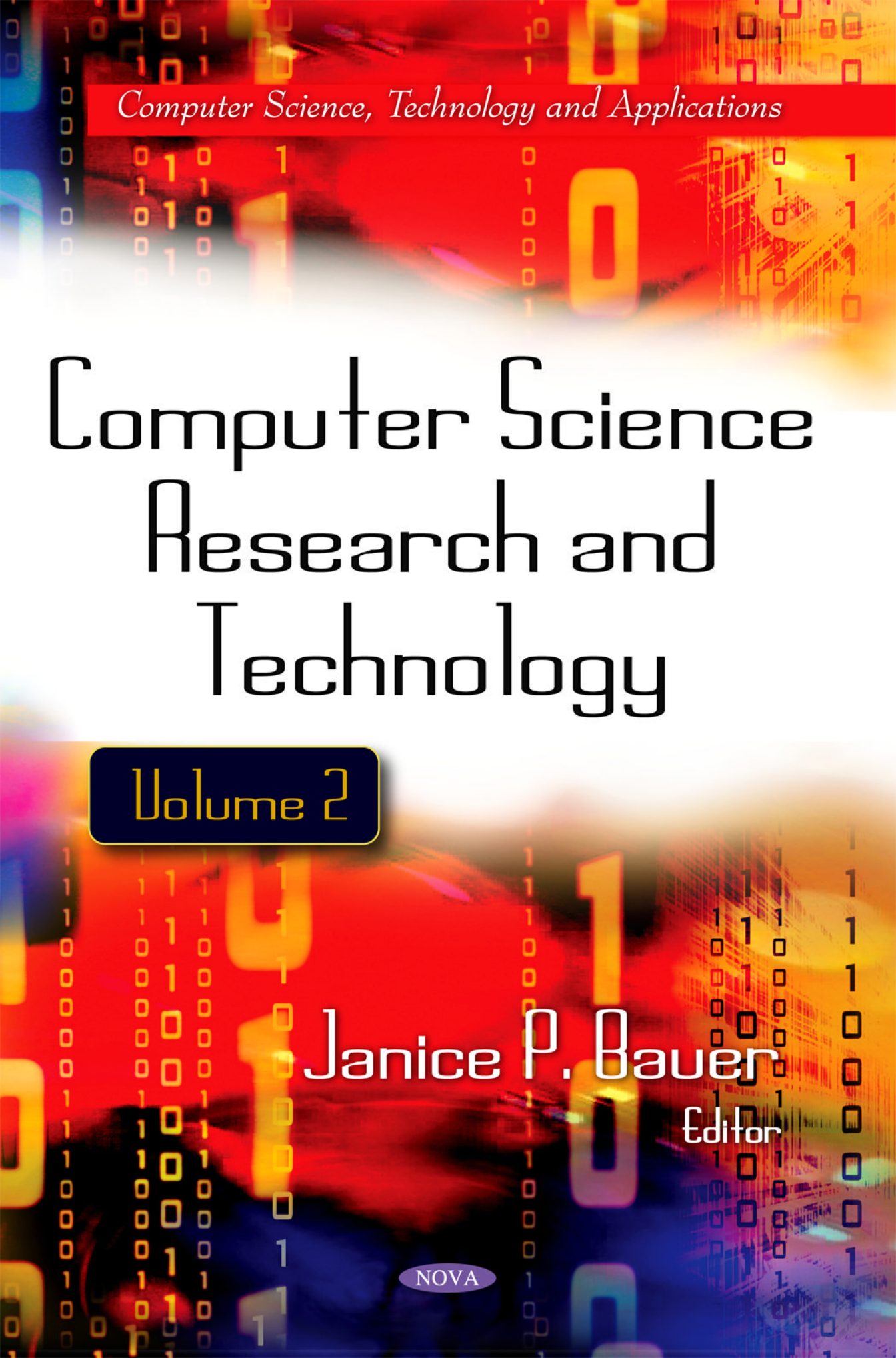

Computer Science, Technology and Applications
Computer Science Research and Technology
Volume 2
Janice P. Bauer
Editor
NOVA

COMPUTER SCIENCE RESEARCH AND TECHNOLOGY. VOLUME 2

COMPUTER SCIENCE, TECHNOLOGY AND APPLICATIONS

Additional books in this series can be found on Nova's website under the Series tab.

Additional E-books in this series can be found on Nova's website under the E-books tab.

COMPUTER SCIENCE, TECHNOLOGY AND APPLICATIONS

COMPUTER SCIENCE RESEARCH AND TECHNOLOGY. VOLUME 2

JANICE P. BAUER
EDITOR

Nova Science Publishers, Inc.
New York

LIBRARY OF CONGRESS CATALOGING-IN-PUBLICATION DATA

ISSN: 2157-8621

ISBN: 978-1-61122-724-6 (eBook)

Published by Nova Science Publishers, Inc. † New York

CONTENTS

PREFACE

This book presents leading-edge research from across the globe in the field of computer science research, technology and applications. Each contribution has been carefully selected for inclusion based on the significance of the research to this fast-moving and diverse field. Some topics included are: Bluetooth Wireless Technology, computer networks; and high performance computing.

Chapter 1 - The need for survivability in optical networks is becoming more and more important because of the huge amount of aggregated traffic carried in an optical network. The main purpose of protection schemes proposed for mesh networks is to provide a network that can restore a failure quickly and efficiently via an automated system. Protection schemes can be divided into two main groups: path-based protection and link-based protection. In this chapter, the authors review and compare the basic protection schemes, unicast protection schemes and multicast protections schemes proposed for optical networks with the mesh topology. The schemes are compared with respect to different parameters such as network complexity and blocking percentage.

Chapter 2 - Cloud computing is an emerging new computing paradigm designed to deliver numerous computing services in ways that have not been experienced before. It is increasingly being viewed as a utility-oriented computing service. There are even speculations that it has the potential to be the 5th utility (after water, electricity, gas and telephony). This computing service is underpinned by a number of existing technologies such as the Internet, virtualization, grid computing and Web services. The flexibility of this computing service approach and its provision in a pay-as-you-go way have opened many possibilities for organizations, of all types, that did not exist before. Among those organizations are those engaged in scientific research. A number of cloud computing services are now emerging to cater for the scientific community, with a potential to bring substantial cost-saving and efficiency advantages. This chapter will highlight those issues and also analyze some of the limitations that are currently associated with this computing service modality.

Chapter 3 - This chapter presents different error control schemes of Bluetooth technology, including the standard error control schemes and custom and adaptive error control strategies proposed in the literature. Bluetooth provides an asynchronous connection-less (ACL) link for data transmission. Bluetooth v2.0 with Enhanced Data Rate (EDR) specification offers three transmission rates, which are basic rate of 1 Mbps (types of DHx and DMx), enhanced data rates of 2 Mbps (EDR2, type of 2-DHx) and 3 Mbps (EDR3, type

of 3-DHx). Bluetooth employs stop and wait ARQ (Automatic Repeat Request), CRC (Cyclic Redundancy Check) and FEC (Forward Error Correction). The payload contains a header of 1 or 2 bytes and a 2-byte cyclic redundancy code (CRC) for checking the integrity of the field after decoding. In DM (Data-Medium rate) packets the payload is encoded with a (15, 10) shortened Hamming code capable of correcting single bit error in each block. Only one fixed packet type is used during the whole Bluetooth transmission period. These characteristics directly affect the throughput and energy efficiency of a Bluetooth network. The choice of the best packet to be used in a transmission depends of the channel conditions. This chapter describes and analyzes some adaptive schemes proposed in the literature for this purpose. Other issue studied by researchers is to apply different FEC schemes in the Bluetooth packets. Some of recent proposals are described in this paper. An analytical framework is presented to analyze the performance of error control schemes. Performance results are obtained through analysis and simulation in Nakagami-m fading channels, indicating the best scheme for each condition.

Chapter 4 - The quantum processor is considered to have the possibility to overcome fundamental limitations of conventional processors, but the influence of energy cost due to the uncertainty principle may prevent speeding up the quantum computation. On the basis of the theorem that the evanescent photon is a superluminal particle, the possibility of a high performance computer system compared with conventional silicon processors has been studied and it can be shown that the energy loss required for computation is much lower than the conventional processor by utilizing evanescent photons. From the high performance capabilities of quantum computation, there are many researchers for explaining the higher performance of human brains, including consciousness. R.Penrose and S.Hameroff have proposed the idea that the human brain can attain high efficient quantum computation by functioning of microtubular structure of neurons. But Tegmark estimated the duration of coherence to be on the order of 10^{-13} seconds, which is far smaller than the one tenth of a second associated with consciousness and it is normally expected that any macroscopic coherence of a quantum state in a warm wet brain to be destroyed almost immediately. Hagen, Hameroff, and Tuszynski have claimed that Tegmark's assumptions should be amended, but the need to maintain macroscopic quantum coherence in a warm wet brain is certainly a serious problem for the Penrose-Hameroff model. From the assumption that the evanescent photon is a superluminal particle, it can be also shown that the microtubule in biological brain can achieve large quantum bits computation compared with the conventional silicon processors, which leads to the high performance computation compared with the conventional processors even at the room temperature, contrary to the Tegmark's calculation result. If this mechanism is true for the brain function, we can obtain the possibility to realize much more efficient computer systems like a human brain by utilizing superluminal evanescent photons for quantum processors.

Chapter 5 - Nowadays many local networks provide high-speed access to end users (e.g., Gigabit Ethernet and optical access networks). Accordingly, each metro network should have a large bandwidth in order to transport the traffic between the local networks and the wide area backbone network. To fulfill this requirement, an *optical metro network* uses optical fibers to connect the local networks to the backbone network via a *metro hub*. In this chapter, the authors describe their recent design of metro hubs called *multi-passive-combiner hubs* [1-2]. A multi-passive-combiner hub is composed of passive combiners and demultiplexers. Within the hub, wavelength channels are assigned to the input-output pairs to fulfill the traffic

requirements while avoiding wavelength conflict. The multi-passive-combiner hub has three major advantages: i) it has low cost and high reliability because it only uses passive optical components, ii) it can efficiently transport non-uniform metro traffic via non-uniform channel assignment within the hub, and iii) it can easily be scaled up to provide more channels for the ever-increasing traffic load.

Chapter 6 - The use of sensors is highly extended in a lot of different environments and applications. Each situation needs a different solution and for that reason the use of a scalable and easily manageable sensor network is a must while applications are getting more and more complex. In a lot of cases the perfect solution is the one based on a wireless sensor network; it gives flexibility, ease of management of the system and expandability. But in order to facilitate interoperability between different sensor manufacturers and to give a transparent and independent interface, the use of a standard is mandatory. This standard system is provided by the IEEE 1451 family of standard protocols. In this project a Bluetooth based sensor network has been implemented using the IEEE 1451 family of standard protocols. The goal of this network is to help on data acquisition from a number of sensors within a car, in order to monitor the wellness of the passengers and improve the safety and comfort.

Chapter 7 - Volunteer computing (VC) is a type of Internet based parallel computing paradigm, which allows any participants on the Internet to contribute their idle computing resources. By exploiting the resources that can be found in millions of Internet computers, VC makes it possible to build a very large and high performance global computing systems with a very low cost. However, VCs still have the problem of guaranteeing computational correctness, due to the inherent unreliability of volunteer participants. In this chapter, the authors fully study the sabotage-tolerance performance of several sabotage-tolerance methods including popular voting method used in real VCs, and reveal their drawbacks in terms of computation time and error rate. Then, based on the conclusion from the comparison study, the authors propose a new job scheduling method for credibility-based voting to efficiently attain the sabotage-tolerance of VCs, i.e. the mathematical expectation of computational error rate is bounded by a given error rate specified as reliability requirement. The key idea for efficient job scheduling is to choose jobs to be executed prior to others considering the important feature that the necessary redundancy for each job may change with time. Extensive simulation has shown that the proposed job scheduling method reduces the overall computation time of VCs, while guaranteeing the reliability requirement.

Chapter 8 - In next-generation networks, packet classification is used to categorize incoming packets into multiple forwarding classes based on pre-defined filters and make information accessible for quality of service or security handling in the network. In the last decade, the technique of packet classification has been widely deployed in various network devices, including routers, firewalls and network intrusion detection systems. To pursue better search performance, numerous algorithms of packet classification have been proposed and optimized for certain filter databases; however, these algorithms may not scale in either storage or speed performance or both for the filter databases with different characteristics. Specifically, packet classification with a large number of filters usually exhibits poor worst-case performance.

In this chapter, the authors improve the performance of packet classification by using multiple hash tables. The existing hash-based algorithms have superior scalability with respect to the required space; however, their search performance may not be comparable to other algorithms. The authors observe the characteristics of the hash-based algorithms and

introduce three hash-based algorithms to improve the performance of packet classification. In the first algorithm, the authors combine two complementary algorithms, Cross-producting and Pruned Tuple Space Search, to make packet classification both fast and scalable. Unlike the existing algorithms whose performance is contingent on the filter database attributes, their algorithm shows better scalability and feasibility. The authors also introduce the procedure of incremental updates. Next, the authors improve the performance of Pruned Tuple Space Search by combining the Aggregate Bit Vector algorithm. The authors also show the related update procedures. Last, the authors present a hash-table reordering algorithm to minimize the number of accessed hash tables with the aid of bitmaps. The authors also use pre-computation to ensure the accuracy of their search procedure. For each algorithm, the authors evaluate the performance with both real and synthetic filter databases of varying sizes and characteristics. The experimental results demonstrate that the new algorithms improve the speed and storage performance simultaneously. Moreover, the results also demonstrate that their schemes are feasible and scalable.

In: Computer Science Research and Technology. Volume 2 ISBN 978-1-61122-073-5
Editor: Janice P. Bauer

Chapter 1

PROTECTION OF UNICAST AND MULTICAST TRAFFIC IN WDM MESH NETWORKS: A SURVEY

Ameneh Daeinabi[a1], Akbar Ghaffarpour Rahbar[a2] and Wail Mardini[b3]

[a]Computer Networks Research Lab, Department of Electrical Engineering, Sahand University of Technology, Tabriz, Iran
[b]Computer Science Department, Faculty of Computer and Information Technology, Jordan University of Science and Technology, Irbid 22110 Jordan

ABSTRACT

The need for survivability in optical networks is becoming more and more important because of the huge amount of aggregated traffic carried in an optical network. The main purpose of protection schemes proposed for mesh networks is to provide a network that can restore a failure quickly and efficiently via an automated system. Protection schemes can be divided into two main groups: path-based protection and link-based protection. In this chapter, we review and compare the basic protection schemes, unicast protection schemes and multicast protections schemes proposed for optical networks with the mesh topology. The schemes are compared with respect to different parameters such as network complexity and blocking percentage.

[1] Email: a_daeinabi@sut.ac.ir.
[2] Email: ghaffarpour@sut.ac.ir.
[3] Email: mardini@just.edu.jo.

Keywords: WDM optical networks, unicast/multicast traffic protection schemes, network complexity, blocking percentage

1. INTRODUCTION

Network and service availability is one of the chief necessities in modern telecommunication networks [1]. In other words, network survivability is an important concern for high-speed WDM optical networks. Due to the ever increasing popularity of bandwidth-intensive applications such as online gaming; video conferencing; e-commerce; and e-learning; network traffic have increased exponentially in recent years [2]. Therefore, all optical networks using Wavelength Division Multiplexing (WDM) technique are fast coming out as the most suitable option for the next generation networks. WDM is a technique that can harness the enormous bandwidth attainable in an optical fiber to satisfy the network requirements. Optical networks using the WDM technology enable the transfer of several Gbps of data on each fiber link in the network. There are two types of failures in WDM networks: link failure and node failure. Respecting improved modern switching devices, link failure is more concern than node failure. Therefore, the single link failure model attracts more attention in the optical survivability research. Therefore, the single link failure can lead to the loss of huge amount of data. Hence, network survivability at the event of link failures is a major concern in WDM networks.

In real world applications, the protection against network failures plays an important role. Network operators would like to ensure that their services are trustworthy and protected against possible cable cuts or failures of various devices used in the network. Since a service or a connection gets unattainable as soon as one of the dedicated networks components becomes unavailable by a failure or imperfection [1], it is necessary to improve suitable protection schemes to avoid or decrease data losses.

Protections schemes have been widely studied in the literature. The complete survey of *p-cycle* schemes has been reviewed in [3] and [4]. The protection schemes for WDM mesh networks and WDM ring networks have been reviewed until 2002 in [5]. Shared protection has been reviewed in [6]. A survey on survivability methods on IP/WDM networks has been provided in [7]. Finally, a limited survey on multicast protection schemes has been presented in [8] and [43]. Although ring is the common physical topology for metro networks, mesh networks are suitable for both metro and wide area networks. In a mesh network, however, survivability is more complex than in a ring topology because of the greater number of routing paths and design decisions that must be taken into consideration [5]. Due to its importance, we provide a comprehensive study on the protection schemes suitable for WDM mesh networks in this chapter. We consider both the protection schemes supporting unicast traffic and the protection schemes supporting multicast traffic.

Before detailing the aforementioned schemes, we would like to define some of the terminologies that will be used throughout this chapter as:

- *Restoration:* restoration is a dynamic mechanism where the backup path will not be setup unless a failure occurs.

- *Recovery time*: the time from the moment of the failure until the traffic is up and running again. The recovery time is dependent mainly on the protection scheme used. The failure notification, the backup path (or backup segment), and the sum of the lengths of the working path (or active segment) are the main factors in determining the recovery time. Note that the recovery time has other components that are independent of protection schemes that are used.
- *Configuration:* it refers how the protection existences are inaugurated to restore working traffic upon failure.
- *Link-disjoint*: two paths are called link-disjoint if they do not pass through any (bidirectional) WDM links in common.
- *Path pair*: it is defined as two paths from a source to each destination node in such a way that paths are link disjoint from each other.
- *Spanning path*: it is defined as a path from a leaf node to any other leaf node in a multicast tree.
- *Intersection node*: if there are two nodes at which a path pair intersects with other exiting path pairs, these two nodes are called intersection nodes.
- *Straddling path*: it is defined as a path between two intersection nodes.
- *Overlapping path*: the sub-path on the path pair from a source to intersection node.
- *Blocking*: if a lightpath for a connection request cannot be found, blocking occurs.
- *Ligthpath:* is an optical channel that spans multiple fiber links to provide a connection between two network nodes.
- *Multicast Capable (MC):* An MC node is a node which has light splitting switch and can forward an incoming message to multiple output channels.
- *Multicast Incapable (MI)*: a node that cannot split the light.
- *Light-tree*: it is a point-to-multipoint extension of a lightpath where its branching nodes are equipped with multicast-capable optical switches [9].
- *Idle-backup edges*: the backup edges which do not carry traffic and are reserved to be used when a failure occurs [10].

Protection against single link failure can be divided into two main groups: link-based protection and path-based protection. Some of their subgroups include Link Protection, p-cycle protection (which also sometimes classified under link protection class), Path Protection, Segment Protection, Partial Path Protection, and Virtual Link Protection. In Link Protection, all connections around a failed link are rerouted. The *p*-cycle is a virtual cycle configured in the spare capacity of a network to protect the working capacity in all the links that have their endpoints on the cycle. Path Protection is a technique in which network resources (bandwidth, buffer, and process capacity) are saved for a unique path protection regarding to a main path. In Segment Protection, a working path is divided into multiple active segments and each working path is protected with a backup segment. In Partial Path Protection, when a link of any path fails, the network can use various protection paths for sending data. In Virtual Link Protection, the protection path is computed for each virtual link, reserved in advance. Sub-Path Protection has two main ideas. First, a large network is divided into several smaller areas by using the open shortest path first (OSPF) routing algorithm. Then, two fiber-disjoint paths are computed for a given connection request.

There are algorithms to protect the unicast traffic detailed in this chapter as: Dynamic hop constrained sub-path protection, Partial SRLG-disjoint shared path protection with differentiated reliability, two-segment algorithm, WDM-shared-non-SRLG Protection, on-line hybrid survivability, survivable algorithm named sub-path protection based on auxiliary virtual topology, Ant-based survivable routing, recursive shared segment protection, Network Coding, and Level of Service Algorithm. Moreover, there is a set of algorithms proposed for protecting multicast traffic, e.g., Optimal Path-Pair based Removing Residual Link removes the residual links in the process of finding the path-pair for a destination node. The other algorithm is Source-Leaf Path based Avoiding Residual Link Algorithm, the Multicast Protection through Spanning Paths algorithm, the Adaptive Share Segment Protection, the Shared Source-Leaf Path based Protection algorithm, the primary tree in spares-splitting WDM networks, Sparse Splitting constrained Multicast Protection, Cross-sharing algorithm, Arc-disjoint tree with differentiated reliability, Partial tree protection, Segment Protection with Most-likely Failure Link, and tree protection scheme with optimized shortest path tree.

In general, we aim to review the recent protection schemes suitable for WDM mesh networks. We focus on the protection schemes supporting unicast traffic and the protection schemes supporting multicast traffic in such networks. Since it may not be possible to compare all the protection schemes with each other due to the inherent implementation, we provide a comparison among different techniques of the same group.

In Section 2, we introduce basic protection techniques. In Section 3, unicast protection schemes are studied. Multicast protection schemes are reviewed in Section 4. To insight on suitable protection schemes, we compare protection schemes with respect to their parameters such as network complexity and blocking percentage in Section 5. The summery of this review is given in Section 6.

2. BASIC PROTECTION TECHNIQUES

The basic protection schemes proposed for WDM mesh networks shall be reviewed in this section. Then, the protection techniques which have been proposed to protect unicast and multicast traffic will be reviewed in future sections.

2.1. Link-Protection

Link protection scheme is based on the routing of all connections around a failed link. When a call request is accepted, the link protection scheme will save network resources for related path protection. Take a notice that two nodes adjoining to a failed link must be connected by a protection path (see Figure1) [11].

When a link failure takes place, the node adjacent to the upstream of the failed link immediately redirects the traffic along the predetermined protection path to the failed link to restore transmission [11]. In fact, end nodes of the failed link are responsible for rerouting of affected traffic around failure without informing the source and destination nodes [12]. In this scheme, the reservation of the protection resource is accounted for each link.

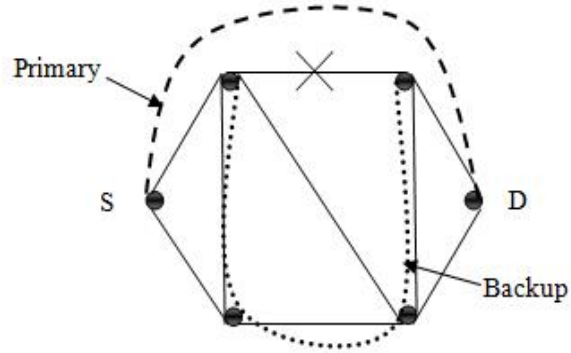


Figure 1. Link protection scheme.

2.1.1. Link-Based Dedicated Protection [13]

The trade of this protection is that the protection capacity is saved for each link and protection resources are not shared among different links. In theory, finding another route for each physical link in a mesh optical network is possible, however in practice, it is hard to find such paths for all the links of an optical network. Moreover, stored capacity in a mesh topology has to be at least twice the working traffic capacity; therefore, it is not so common in an optical network to use this approach (i.e., using two parallel fibers for each physical link).

2.1.2. Link-Based Shared Protection [13]

In this scheme, various links, which may not fail simultaneously, are able to share the same resources for protection. Thus, shared protection can reduce the total amount of resource required for protection. But in this scheme, extra time is necessary for signaling and configuration processes.

2.1.3. Virtual-Link Protection [13]

This scheme can be used in optical networks that permit end-to-end traffic to move on a sequence of multiple working paths. The protection path is computed for each virtual link, which can be reserved in advance, so a very high efficient protection assurance with the minimum amount of resource for protection is anticipated. Virtual protection needs complex computations because of the failure spreading issues. When a failure occurs, the end nodes of the failed virtual links begin the restoration process. Two virtual link-based protections can be implemented as:

- *virtual link-based shared protection*: signaling and configuration processes must be completed before the source node of the failed virtual link is permitted to switch the traffic to the protection path.
- *virtual link-based dedicated protection*: the waiting time that should be spent for completion of the configuration process can be omitted by performing pre-configuration. Then, traffic can be switched to the protection virtual link immediately.

2.1.4. The p-Cycle Protection

The *p*-cycle is a virtual cycle in a network and can protect working capacity of the link that has both end points on the *p*-cycle. The *p*-cycle protection scheme uses additional capacity in cycles to protect the links. Usually, different amount of capacity is needed on all the links of a *p*-cycle. The capacity is pre-configured in such a way that when a link fails, only the end nodes of the failed link need to do rerouting. Hence, no signaling is required. The *p*-cycle protects two types of links, *on-cycle* links and *straddling* (chord) links. In Figure2, the thick solid lines indicate the pre-configured capacity of *p*-cycle [14].

In on-cycle protection, it is assumed that there is another way around a cycle when a failure occurs in order to reroute the connections over the other way. In a straddling link, the end-nodes are on the cycle but the link is not on the cycle, and the cycle has two routes between the end-nodes of the failed link. The *p*-cycle can hence protect twice the pre-configured capacity of the *p*-cycle [14]. A link may be protected by several *p*-cycles, i.e. if a link fails, the connections using that link may be protected by rerouting them along several different *p*-cycles protecting that link [14].

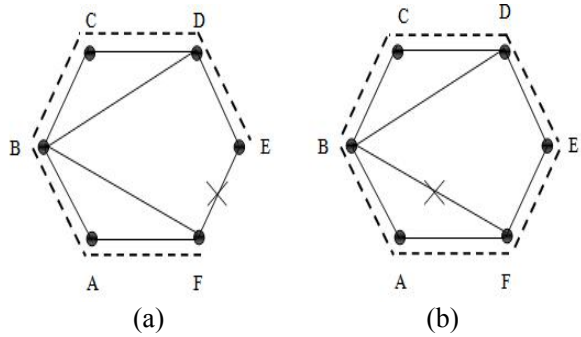


Figure 2. p-cycles protection scheme: a)on-cycle link protection. b)straddling link protection.

The optimal solution is usually hard to find because we need to enumerate all the cycles of a graph. In a planar graph that can be applied for many real networks, the scheme using special kind of cycles would be very fast to find very good solution [15]. Pre-configuration means that the configuration is done before happening a failure [16].

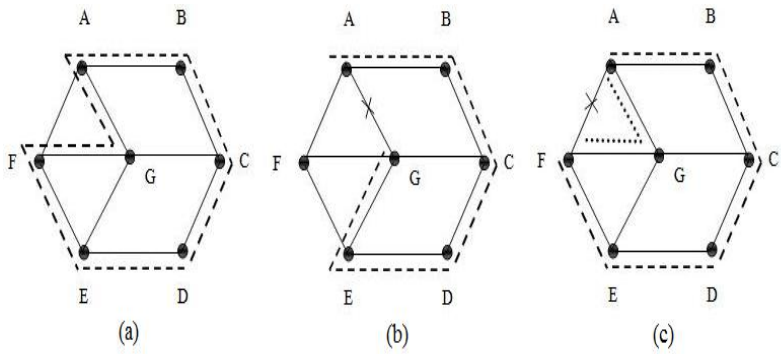


Figure 3. Protection principle of p-cycles for link protection.

Figure3 depicts the protection principle of *p*-cycle for link protection. The *p*-cycle in Figure3(a) is pre-configured as a closed connection on the cycle A-G-F-E-D-C-B-A. On-cycle links are shown in Figure3(b). Upon failure of the on-cycle link A-F, the *p*-cycle offers protection by the route on the remainder of the cycle (G- F-E-D-C-B-A). Figure3 (c) shows the protection of such a link (F-A) on straddling links. Two protection routes can be provided for straddling links, i.e., the routes A-B-C-D-E-F and A-G-F in the example [16]. In practice, one can protect two working capacity units of straddling links.

2.1.5. Segment Protection

In this scheme, working path is divided into multiple active segments and each working path is protected with a backup segment (instead of protecting the whole working path in the path protection scheme). This scheme is failure-dependent path protection, but the rerouted traffic only needs to be from the node of the backup segment that protects the failed link [17]. To achieve bandwidth efficiency in links under path or segment protection schemes, backup bandwidth allocated to different backup segments or backup paths can be shared. A tradeoff between time and network utilization (or blocking execution) may be obtained using this scheme [18].

2.2. Path-Protection

In path-protection, network resources (such as bandwidth, buffer, and process capacity) are saved for a unique path protection for the primary working path. Thereat, there is no anticipation which link along the primary working path will fail; the system allocates a protection path, which is link-disjoint from the primary path (Figure4). Therefore, the primary path does not have any subscriber link with its relevant path protection [11].

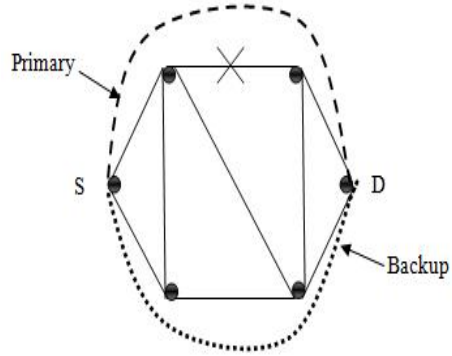


Figure 4. Path protection scheme.

When a link fails, the source and destination nodes on the failed link are informed, and then the source node can redirect the failed traffic on the protection path (i.e., the communication is switched to the protection path) [11]. As soon as a failure takes place, an error warning is sent to the source and destination of each affected connection. Each pair of the source and destination nodes acts for communication switching from the main path to the backup path [13].

2.2.1. Path-Based Dedicated Protection [13]

In this scheme, one dedicated protection path, which is link-disjoint from the working path, is reserved for each working path in advance. Since a protection resource is dedicated for each protection path, the configuration of equipments along the path can be performed in advance. This pre-configuration process can reduce restoration time. When a failure takes place, the source node can switch the traffic to the protection path without consuming extra time waiting for signaling and configuration processes. Advantage of this scheme is its simple design; however, its disadvantage is that it eliminates sharing benefits, which results in a large capacity needed for protection.

2.2.1.1. The 1+1 Protection

This is a kind of Path-based dedicated protection scheme in which there are two links or nodes with separated path for ensuring of sending information connectivity service in case of having defect in network elements. The signal is sent over both working and backup paths at the same time. Under a failure situation, a receiving terminal chooses the best signal according to local information about the signal quality. Line efficiency is supported with allocating protection channel for each working channel [1]. Figure 5 illustrates a configuration example for two demands, F-N and F-H. Both demands are routed along two node-disjoint paths. It is obvious that the required minimum capacity is twice the amount of network capacity needed to transport the traffic demand as in the unprotected case [11].

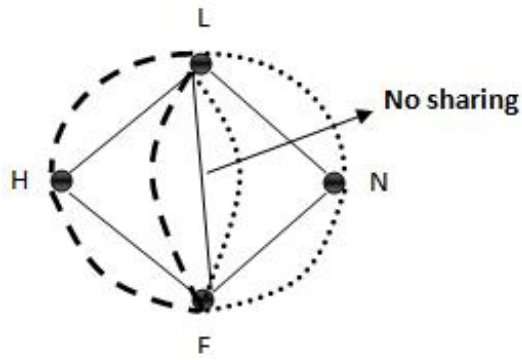


Figure 5. Example of 1+1 protection.

2.2.2. Path-Based Shared Protection

The essence of this protection scheme is the sharing of protection resources among working paths. However, a protection path cannot be shared among links that may fail simultaneously. In other words, if working paths are separate, it is possible to share backup resources. Usually, the purpose of this scheme is to choose the protection path where the amount of the reserved capacity for protection is minimized. Since equipments like optical switches are shared amongst several working paths, the equipment configuration for protection is commonly performed after a failure takes place. End nodes of a failed link can detect the failure and send the signaling message to each network node in protection path to adjust process. Then, the source node can switch its current traffic on the protection path. In shared path protection, the backup paths are predefined and capacity (wavelength) is reserved,

but it is not used. When a particular main path fails, traffic is switched to the saved backup path for coping with the failure. Note that the computation complexity has a direct relationship with failure spreading. Failure spreading is the condition in which the failure of a fiber can cause several upper layer failures [13]. In Figure6, for instance, consider that a link failure occurs, where this failure will not affect the working (active) path AP1 and the working path AP2 at the same time. Suppose that w_1 and w_2 are required bandwidths on AP1 and AP2, respectively. Corresponding backup paths can share bandwidth on common link *e*. This means that the amount of the reserved backup bandwidth on link *e* (w_1, w_2) equals the maximum of w_1 and w_2, not w_1+w_2 [17, 19].

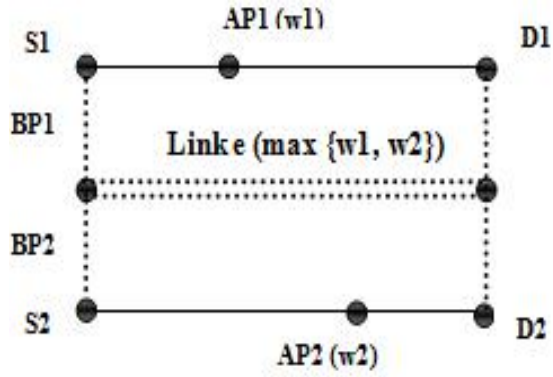


Figure 6. Shared path protection scheme.

2.2.2.1. The 1: N Protection

The 1: N protection provides one protection channel that protects N working channels, where N ≥ 1. In 1: N protection, there are still multiple fibers between a source and a destination nodes but traffic is transmitted only over the working channels, while the protection channel is kept free until a working channel fails. The failure of the working channel causes the protection channel to automatically take over and restore data flow to the network. Note that if multiple working channels fail simultaneously, traffic from only one of the failed working channels is switched to the protection channel.

2.2.3. Partial Path Protection

In Partial Path Protection, when a link of any path fails, the network can use various protection paths for sending data. Partial path protection re-uses operational segments of the main path in path protection [11]. A collection of one or more backup paths of a partial path protection is used for each working link so that the collection of backup paths collectively protects all channels on the working path, where each protective path should be link-disjoint only from the link which it protects [20]. In this way, the system can reserve protection resources during adjusting the main path. The main distinction with path protection is that it specifies a specific path protection for each link along the main path (i.e., each link of the main path is protected). In case of a failure, traffic is rerouted along the protection path corresponding to the failed link.

Figure7 shows a primary path 1-2-3-6-4-5, where node 1 is the source node and node 5 is the destination node. As depicted in Table 1, the system using partial path protection takes 1-3-6-4-5 as the protection path against the failure of link (1, 2). Similarly, the network allocates 1-2-3-4-5 to protect against the failure of (3, 6). Each of these protection paths needs to be link-disjoint only from the link that it protects.

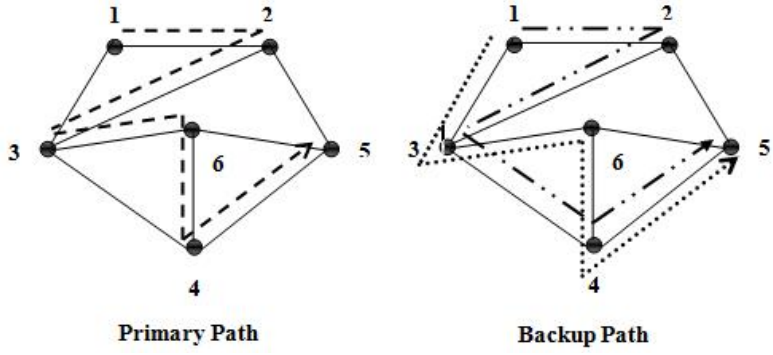


Figure 7. An example for partial path protection scheme.

Consider the network in Figure8 and let the network have initially no connections. The network now serves two call requests (1,4) and (4,5) in sequence. Table 2 shows the resource assignments for primary and protection paths under the path protection and partial path protection respectively. As Table 2 shows, the two primary paths 1-6-3-4 and 4-5 are completely link-disjoint from each other. By protection sharing, the system reserves only one wavelength for protection on link (2, 5), thus improving the utilization of network resources. In path protection, a wavelength on this link is allocated to protect link (1, 6), (6, 3) and (3, 4), while in partial path protection the wavelength protects only the link (1, 6). Hence, under the partial path protection, this wavelength can be shared by a future call whose primary path includes link (6, 3) and (3, 4), but cannot be shared using the path protection [11].

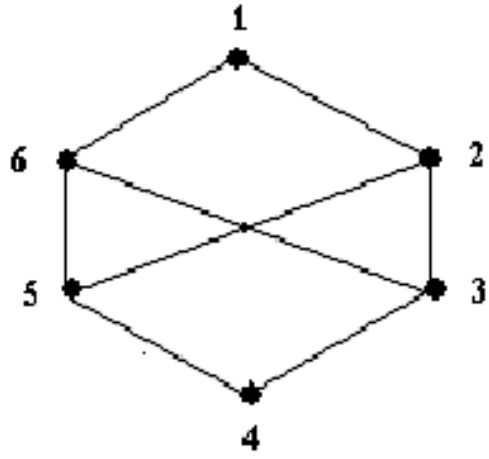


Figure 8. An example network for illustration of the partial path protection and path protection schemes in protection sharing.

Table 1. Illustration of protection path for the primary path shown in Figure7

Link on Primary Path 1-2-3-6-4-5	Corresponding Protection Path
(1,2)	1-3-6-4-5
(3,6)	1-2-3-4-5
(6,4)	1-2-3-4-5

Table 2. Resource allocation for source destination pair (1,4) and (4, 5) of the network in Figure8

	SD Pair	Primary Path	Protection Path (protected link)
Path Protection Scheme	(1,4)	1-6-3-4	1-2-5-4 (1,6) 1-2-5-4 (6,3) 1-2-5-4 (3,4)
	(4,5)	4-5	4-3-2-5 (4-5)
Partial Path Protection Scheme	(1,4)	1-6-3-4	1-2-3-4 (1,6) 1-6-5-4 (6,3) 1-2-5-4 (3,4)
	(4,5)	4-5	4-3-2-5 (4,5)

2.2.4. Sub-Path Protection [21]

This scheme has two main steps: (1) a large network is divided into several smaller areas using the open shortest path first (OSPF) routing algorithm; and then (2) two fiber-disjoint paths p_w and p_b are computed for a given connection request d in such a way that p_w and p_b enter (or exit) an area from the same ingress (or egress) area border router (ARB) if d is an inter-area connection, and they remain in the same area if d is an intra-area connection.

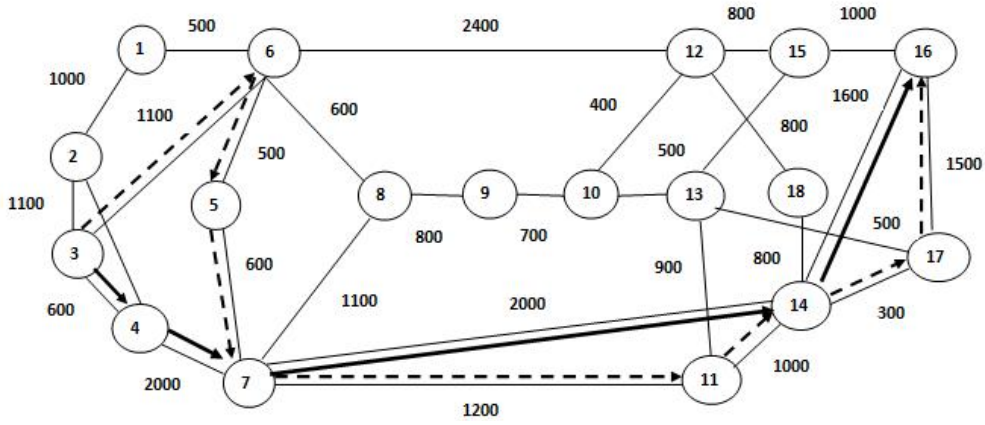


Figure 9. An example 18-nodes nationwide network where each cloud denotes an area.

Consider an inter-area connection between node-pair (3,16) as shown in Figure9. The working path p_w is <3,4,7,14,16>, and the backup path p_b is <3,6,5,7,11,14,17,16>. The paths p_w and p_b exit area 1 (and enter area2) from the same ABR 7 and exit area 2 (and enter area 3) from the same ABR 14. The ABRs 7 and 14 segment p_w and p_b into three sub-paths as: <3,4,7> (p_{w1}), <7,14> (p_{w2}), and <14,16> (p_{w3}) for p_w ; <3,6,5,7> (p_{b1}), <7,11,14> (p_{b2}), and <14,17,16> (p_{b3}) for p_b [13]. Each (p_{wi} , p_{bi}) pair corresponds to the working and backup paths

of some intra-area traffic. When fiber link <*i,j*> along p_w fails, instead of shutting down the entire working path p_w and switching the traffic to p_b, this scheme only turns down the sub-path p_{wk} and switches the traffic to p_{bk}. Therefore, other sub-paths that do not traverse fiber link <*i,j*> along p_w are not affected. Because of the independence of sub-paths, sub-path protection can survive from up to α failures as long as there is at most one failure per area, where α is the number of areas.

3. UNICAST TRAFFIC PROTECTION TECHNIQUE

Ten major protection schemes (see Figure10) that support unicast traffic in WDM mesh networks are reviewed in this section.

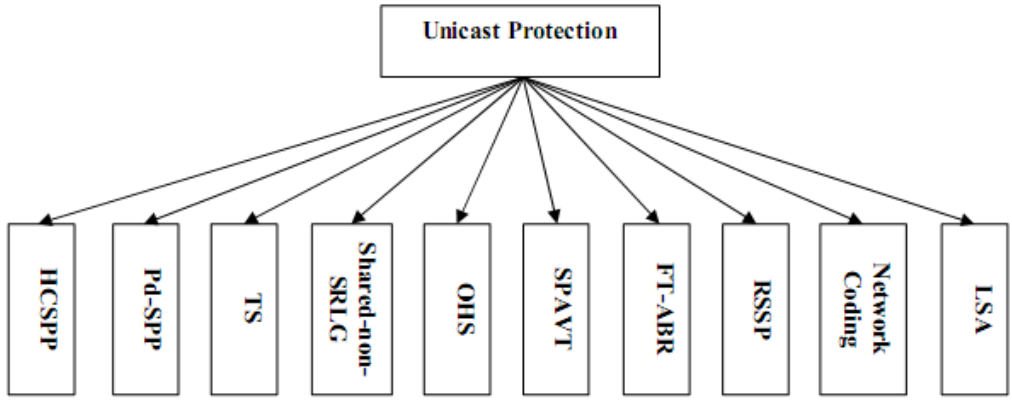


Figure 10. Reviewed unicast protection algorithms.

3.1. Dynamic Hop Constrained Sub-Path Protection (HCSPP) [22]

The HCSPP can improve cooperation between the blocking probability and failure recovery time. In this way, the straddling path and hop count are used to subdivide a protection cycle. When a request connection arrives, in the first step, the primary and the end-to-end backup paths should be calculated. After computing the costs of primary path and its corresponding end-to-end backup path and also forming the cycle, the hop counts of the cycle is computed. If the hop count is lower than a threshold, then the corresponding wavelength resources are recorded. For every node along primary and end-to-end backup paths (excluding the source and destination), the preliminary straddling path which should be link-disjoint with primary and end-to-end backup paths is computed. The shortest hop count straddling paths which start from each node are selected and added to a list. Then, the list is sorted based on an ascending order of node numbers. Therefore, the protection cycle which is formed through the primary path and end-to-end backup path is divided by the selected straddling paths from the list. Finally, redundant straddling paths are removed.

3.2. Partial SRLG-Disjoint Shared Path Protection with Differentiated Reliability (Pd-SPP)[23]

In order to improve the network resources utilization, [23] uses shared risk link group (SRLG) -disjoint, the *k*-shortest paths algorithm and differentiated reliability. The shared risk link group is a group of network links in which a common physical resource (such as fiber, cable) is shared among them, and any link failure within the group will cause the failure of all links in that group. The information on SRLGs is manually attained based on the knowledge of the physical fiber plant of the network by the network operator. Note the reliability is the probability that a system or connection will operate correctly in a period of time [23]. For the source node *s*, destination node *d*, bandwidth request *b*, and requested reliability *r*, the procedure of Pd-SPP is executed as follows:

When a connection request arrives, a temporary graph is first created using the links that their free bandwidths are more than *b*. Second, the *k*-shortest path algorithm is executed until *k* candidate working paths are found. Then, for each candidate working path, all the links and nodes belonging to candidate working paths are added to the temporary graph and the bandwidth of all added links are recorded. The Dijkstra's algorithm is run to compute the shortest path. This process is executed until the path pair is found. In order to satisfy the reliability *r*, Eq.(1) should be verified:

$$F_{b/w} \leq \frac{1-R}{1-\prod_{l_i \in P_{wp}} r_i} \qquad (1)$$

where $F_{b/w}$ is the conditional failure probability of the backup path when the working path (P_{wp}) fails, *R* represents the requested reliability of the connection, l_i and r_i are the wavelength link *i* and the reliability of l_i, respectively.

After determining the path pair, bandwidth resources are assigned to the path, where *R* free bandwidths are allocated to each link on a working path. Moreover, for each link belonging to a backup path, the number of free bandwidths that can be assigned to the backup path is determined based on the shareable backup bandwidth. Finally, the connection is established. If no candidate path pair is found, the connection request is blocked. In order to release the working and backup paths, a reversed process of the call setup is executed.

3.3. Two-Segment (TS) Protection [15]

In order to prevent "traps" in survivable WDM mesh networks, the new segment protection is suggested in [15]. The trap problem occurs when an SRLG-disjoint backup path cannot be found for an active path [15]. The main idea of TS is that an active path should be protected through two backup segments if there is a trap problem for the given active path. The procedure of the TS algorithm is as follows:

- For a connection request and its path (*p*) from source node *s* to destination node *d* and intermediate nodes n_i (where i=1,…,k), first the links belonging to *p* as well as other links that are not SRLG-diverse with *p* should be removed (i.e., the links that use the same SRLG with *p*).

- To form the auxiliary graph (G), the reverse link of each directed link belonging to p is added to G and then the costs of them are set to 0.
- For each active segment (S) of node pair (s, n_i) where i=1,…,k, first the links belonging to S as well as other links that are not SRLG-diverse with S are removed from G. Then, the SRLG-disjoint backup segment (BS) for S is attained through the Dijkstra's algorithm.
- A directed link is added from s to n_i. The costs of links of the backup segment (BS) are equal to the link cost.
- The same procedure is executed for each active segment (S) of node pair (n_i, d) where i=1,…,k.
- Finally, for each node pair (s, d), the shortest path in G is derived through the shortest path algorithm and corresponding backup segments belonging to the links of the shortest path are recorded.

3.4. WDM-Shared Non-SRLG Protection [24]

This algorithm determines whether an established backup path can share resources based on the Shared Risk Link Group (SRLG) or not. Note that the maximum value of the SRLG-percentage should be considered because it is a unidirectional property, where the constraint of the SRLG–percentage shows the maximum percentage of the primary links that are shared.

In this algorithm, there are three steps: the link weight scaling; the shortest path computation; and the wavelength channel reservation. These steps are executed twice: one time to compute a primary path and another time to calculate a backup path. Computing the primary path is as follows:

- The weight of a link is set to infinity, if the link does not have available resources or has failed. Otherwise, the weight scales based on the percentage of usages.
- A shortest path algorithm such as Dijkstra is used to calculate the path in the weighted graph.
- The entire path is traversed and it is determined which wavelength channel should be used in each link.
- In order to establish the backup path, the following process is run:
- First, the weights of the edges used by the primary path must be set to infinity to establish the backup path.
- The shortest path algorithm is run in such a way that the primary links are not considered as candidates for backup path because the primary and backup paths must be link-disjoint.
- The third step is executed similar to the third step of computing the primary path procedure, only with two tunings. This means that in addition to its regular routines, it is possible to share a wavelength channel already reserved to another backup path when backup sharing is used. If the backup sharing is used and in this case a failure happens, the backup availability test must verify other connections associated with this backup path.

3.5. On-Line Hybrid Survivability (OHS) [25]

OHS offers a new approach for dynamic connection requests with multiple levels of service such that for a new arriving connection request, the cost of the link according to the current state of the network is adjusted immediately, and then, the primary and backup paths are computed. The main aim is to improve network resource utilization and blocking ratio. Three levels of services are considered in OHS: the highest level of service is represented by level 2. In this level, when a connection request arrives at the network, primary and shared backup paths are elected. Level 1 demonstrates the middle level of service, where the primary path does not share any resources with the backup paths of level 2. The lowest level of service is shown by level 0, where no survivable mechanisms are provided for the class of connections.

When a connection request arrives, the level of the connection is first verified. If the connection request belongs to level 2, according to Eqs. (2)-(3) the link cost is tuned and then a least primary path is computed. If the primary path is found successfully, the link cost of the backup path is tuned with respect to Eqs. (4)-(6), and then, a least-cost path is allocated as a backup path. Finally, the selected primary and backup paths and their allocated wavelengths are saved. If primary or backup paths are not found, their connection requests are blocked and the allocated network resources are released. For a connection request belonging to level 1, the link cost is computed according to Eqs. (2)-(3) and for a connection request belonging to level 0 the link cost is calculated according to Eqs. (4)-(6). Then, for both levels 1 and 0, the least primary path is determined and the found path and assigned wavelength are recorded. If the primary path cannot be found, the connection request is blocked.

$$C_e = +\infty \; if \; F_e = 0 \tag{2}$$

$$C_e = \frac{(R_e + W_e)}{F_e} \; if \; F_e > 0 \tag{3}$$

$$C'_e = \delta \; if \; max_{e'} = R_{e'} \tag{4}$$

$$C'_e = \frac{max_{e'} - R_{e'}}{F'_e + R_{e'} - max_{e'}} \; if \; F'_e < max_{e'} < F'_e + R_{e'} \tag{5}$$

$$C'_e = +\infty \; if \; max_{e'} \geq F'_e + R_{e'} \tag{6}$$

where e and e' are unidirectional links. The parameter F_e represents free capacity, R_e shows reserved capacity which can be shared by some backup paths, C represents the set of existent connections in the network, δ is a sufficient small positive constant say 0.01 or 0.001, and W_e is working capacity taken by some primary paths and cannot be used for any other purposes until the corresponding primary path is released. The parameter $max_{e'}$ denotes the required backup resources on the unidirectional link-cost of each unidirectional link e' for the backup path of connection request passing over e' [25].

3.6. Sub-Path Protection Based on Auxiliary Virtual Topology (SPAVT) [26]

In order to solve the problem of the protection switching time, SPAVT has been proposed, where the protection switching time denotes the time taken from the instant a link fails to the instant the backup path of a connection traversing the failed link is enabled [26]. In this method, the virtual topology is formed based on the multiple pairs of working and backup paths for each node pair in the network using an off-line mode. After receiving a connection request, the Dijkstra's algorithm is run once within the virtual topology to find a virtual route. Finally, the optimal pair of sub-paths is selected. The SPVAT algorithm is executed as follows:

- The set of primary paths between each node pair (x, y) is created through the k-shortest path algorithm.
- The set of backup paths for each node pair (x, y) is computed by k-shortest path algorithm. Note the backup paths are link-disjoint with the corresponding primary path.
- The virtual topology T is made. When a connection request $R(m, n)$ arrives, a virtual route such as m'-a'-b'-n' is computed. Note that (m' , n') is virtual node pair corresponding to node pair (m, n) in T. Moreover, á and ɓ are virtual nodes. If an appropriate route is found, the sub-primary paths in the set of primary paths between each node pair (m', a'), (a' , b') and (b' , n') on the virtual route m' - a' - b' - n' are selected, considering least working resources. Moreover, sub-backup paths are elected similar to the way explained for selecting sub-primary path.
- The primary path, backup path, and consumed resources are recorded for the connection request and then the network state is updated. The system then waits for a new connection request arrival. If no appropriate path is found, the request is blocked and the network state is updated.

3.7. Fault-Tolerate Ant-Based Survivable Routing with Shared Backup Path (FT-ABR) [27]

This approach is based on the ant-colony behavior and focuses on dynamic routing and wavelength assignment (RWA) problem. The Dynamic RWA is performed online when connection requests arrive. This approach assumes that ant-based mobile agents run in a separated control plane that is carried out in a packet switching network with the same topology as optical network, or in the optical domain where control data is transported on a dedicated wavelength [27]. Moreover, each network node has two tables: a routing table with a set of feasible protection cycles between source-destination nodes, and a pheromone table for mobile agents. In order to increase the network performance, low-cost cycles are reported to each network node through mobile agents.

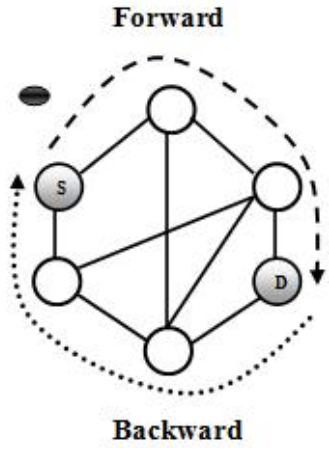


Figure11. Architecture of a wavelength-routed network.

Every t time, in the forward ant step, the ant with a given probability ρ starts moving from each node to the randomly selected destination. An ant stochastically determines its next hop according to its selecting probabilities in the pheromone table. Each ant is considered to be a mobile agent. Then, the forward ant collects information on its travel and updates routing tables on visited nodes. Moreover, forward ant updates the pheromone table with collected information. The moving nature of agents and the update of the pheromone table makes ant-based mobile agents tend to follow paths with shorter length and larger number of free wavelengths. After the ant reaches the destination, it moves back to the source node based on the same forward rules. Note that backward ant should check its path not to move on the visited links so that all the visited nodes will formulate a cycle. If the ant cannot find a node to move, it will be killed. After that the ant arrives at the source node, it reports the cycle to the routing table and then is killed. In order to select a minimum cost link, the cost of primary path and the cost of backup path are first computed for each wavelength. Then, a set of cycles with the minimum cost of primary path are selected among the cycles with at least one wavelength on which cost of the backup path is not infinity. Finally, the cycle with minimum cost of link is selected.

3.8. Recursive Shared Segment Protection (RSSP) [28]

In order to achieve better recovery time and resource utilization, [28] has proposed a recursive shared segment protection (RSSP) that calculates the backup segments through a recursive method and selects the optimized of them to partition the working path. In RSSP, there are several restricts: each link and node belonging to a working path can only be a part of one working segment or at most two working segments. Moreover, any working segment should not be a subset of any other working segment. Finally, the hop count parameter of each backup segment should be limited by a threshold value, where the threshold is a positive integer represented by H_c. The RSSP is executed as follows:

- When a connection request arrives, the working path should first be computed. For this purpose, the cost of each link is tuned based on Eq.(7) and then through the Dijkstra's algorithm, the minimal cost path is considered as working path from the source node to the destination node. The cost function cwp_i for finding the working path with the requested bandwidth (RB) on i-th link is defined as:

$$\mathrm{cwp_i} = \begin{cases} \mathrm{c_i} & \mathrm{f_i} \geq \mathrm{RB} \\ +\infty & \mathrm{f_i} < RB \end{cases} \quad (7)$$

where ci is the basic cost of i-th link and can be a constant or a variable number. Moreover, fi is the free capacity.

- After finding the working path, the first working and backup segments should be calculated. Therefore, the beginning node is considered as the first node and also any node among source and destination is considered as an end node. Then, for every candidate working segment, the cost of each link in the network is tuned according to Eq.(8). The cost function cbsi for computing the backup segment with the requested bandwidth (RB) on i-th link is computed as

$$\mathrm{cbs_i} = \begin{cases} \varepsilon & \mathrm{RB} \leq \mathrm{sh_i} \\ \varepsilon + \alpha.\frac{\mathrm{RB}-\mathrm{sh_i}}{\mathrm{fi}} & \mathrm{sh_i} < RB \leq \mathrm{sh_i} + \mathrm{f_i} \\ +\infty & \mathrm{sh_i} + \mathrm{f_i} < RB \end{cases} \quad (8)$$

where ε is a sufficient small positive constant say 0.01 or 0.001, and α is a positive constant parameter. The parameter shi is shareable capacity, which is the capacity reserved by some working segments and is shared by this working segment, where the "some working segments" should be link-disjoint with this working segment [28].

- The minimal cost path from the first node and each end node is found and denoted as the corresponding candidate backup segment if its hop count is lower or equal to Hc.
- The candidate backup segment with the highest priority is considered as the first backup segment. If the first backup segment cannot be found, the connection request is blocked.
- If the selected end node of the first backup segment equals destination, the first backup segment can cover the whole working path and then the bandwidth resources are assigned to the first pair of working and backup segments. Therefore, the algorithm is over.
- The algorithm should continue if the first backup segment cannot protect the whole working path and another required working and backup segment has to be determined.

3.9. Network Coding [29]

This refers to performing linear coding operations on traffic carried by the network at intermediate network nodes. In network coding, a node receives information from all or some

of its input links, encodes them, and then forwards the information via multicasting or broadcasting [29]. Using this scheme, the network capacity can be enhanced. For the purpose of data recovery, *p*-cycle will be used to deliver the combinations of data units to destination. The 1+N protection against single link failures has three steps that are detailed as follows:

A. The p_Cycle Construction and Node Assignment Procedure to Cycles

First, a full *p*-cycle is found. The basic idea for receiving a second copy of data by a node is to receive the signal via opposite directions; therefore, a full *p*-cycle is divided into two unidirectional half *p*-cycles in opposite directions. These two *p*-cycles do not pass through the same links, but must pass through the nodes in the same order. Then, two classes of communication nodes should be selected in such a way that a node in one class communicates with a node in another class. The sequence of nodes in each class is arbitrarily selected. Nodes in one class have counter-clockwise direction and in another class have clockwise direction. One of the two half *p*-cycles is a clockwise half *p*-cycle, and the other is a counter-clockwise half *p*-cycle. They are used as follows:

a. A half *p*-cycle in the clockwise direction: On this half cycle, every node in each class generates data units, encodes, and then transmits them. These encoded data are decoded and removed by the corresponding receiver in two classes.
b. A half *p*-cycle in the counter-clockwise direction: Data units received on the primary working paths in each class are encoded and transmitted by the nodes in every class. The encoded data units are decoded and removed by the corresponding transmitters in each class.

Note that encoding and decoding operations are simple module 2 addition operations applied on data units that are transmitted and received on the same cycles. Transmissions take place in rounds, i.e., encoded and transmitted data units on the *p*-cycle must belong to the same round. A round can be started by one node, and then followed by other nodes. All nodes in each class must keep track of the round number.

B. Encoding Operations

For every node in each class the network encoding operation is executed as:

a. The node will add the following data units to the signal received on the clockwise direction:
   - Data unit generated by one node.
   - Data unit received on the primary path from a node on another class.
b. The node will add the following data units to the signal received on counter-clockwise direction and will transmit the result on the outgoing link in the counter-clockwise direction:
   - Data units transmitted in an earlier round.
   - Data units received on the primary path from a node in another class.

C. Recovery from Failure

If a link on the path between two nodes (where each node belongs to one of the two classes) fails, destination node will not receive data on the primary path. However, it can recover data by using results of the encoding operation.

3.10. Level of Service Algorithm (LSA) [30]

In priority-based routing scheme, traffic is divided into high-priority (HP) traffic request and low-priority (LP) traffic request. The HP traffic accepts the shared backup path protection and its backup path can share the resource with the risk-disjoint HP traffic path. Moreover, the HP traffic's backup path can use the working resources of LP traffic. The LP traffic will adopt the restoration scheme, and there is no backup resources allocated to this traffic. When failure occurs, the LP traffic immediately starts searching for a recovery path.

The procedure steps of the heuristic algorithm, called Level of Service Algorithm (LSA), are presented as follows: when an HP traffic request arrives, a primary path is computed according to the standard shortest path algorithm. Then, a risk-disjoint backup path is calculated through the standard shortest path algorithm, the routes are recorded and the wavelength is allocated. Moreover, the network state is updated. If an LP traffic request arrives, a primary path is computed according to the standard shortest path algorithm. Then, the routes are recorded and the relevant wavelength is allocated. Moreover, the network state is updated. If the algorithm cannot find any primary or backup paths, the connection request will be blocked.

4. MULTICAST TRAFFIC PROTECTION SCHEMES

Multicast-based services are popular applications in WDM networks [31]. Since a single fiber cut may cause much more significant breakages to a light tree than an individual lightpath, therefore, protection of a multicast session is very important.

Multicasting involves the propagation of information from one source to multiple destinations at the same time [32]. A point-to-multipoint connection from a source to multiple destination nodes is defined as a light-tree to maintain a multicast session on WDM networks and it is the generalized expansion of a lightpath [31]. An efficient approach for multicasting is to establish a multicast tree, which contains the source node as the root and all the destination nodes as the leaves or intermediate nodes [32]. Network nodes must be multicasting-capable optical cross-connect switches (MC-OXCs), where these switches split the received optical signal into multiple paths in order to support multicasting in a WDM network. Note that each path of the optical signal is switched to a desired output port [32]. In the following, major techniques to protect multicast traffic are reviewed (see Figure12).

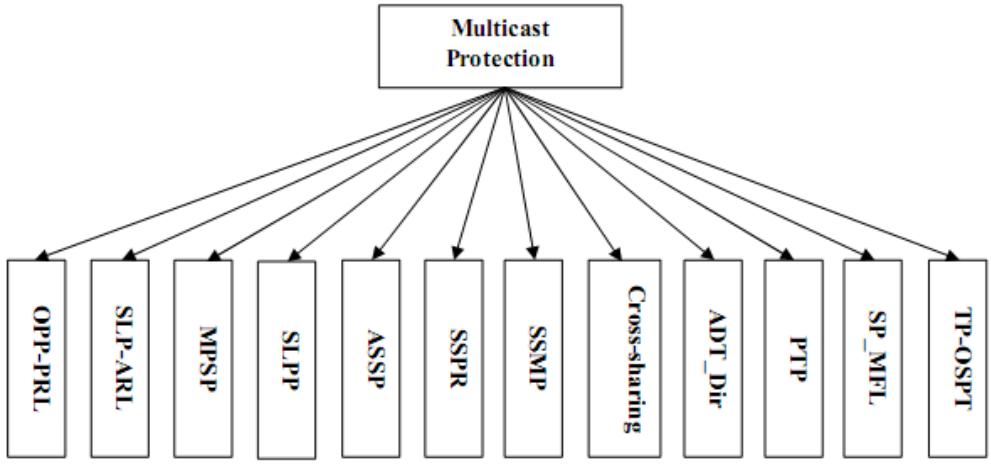


Figure 12. Reviewed multicast protection algorithms.

4.1. Dynamic Multicast Protection Schemes

Two algorithms are suggested to reduce the session blocking probability and to decrease the usage of network resources by removing or avoiding residual links in the topology that consists of "light-tree" and its backup path.

4.1.1. Optimal Path-Pair Based Removing Residual Link (OPP-PRL) [32]

The OPP-PRL algorithm is derived from the Optimal Path-Pair based Shared Disjoint Path (OPP-SDP). The OPP-SDP algorithm computes an optimal path pair between a source and destination pair. In the OPP-PRL algorithm, a link-disjoint path pair is found for each destination node. Using this algorithm, residual links between the source and destination can be removed. In order to minimize additional costs and to increase sharing of new path pair with those found previously, the OPP-SDP algorithm updates the cost to zero for the links along the found path. Moreover, this algorithm decreases the usage of network resources by removing the overlapping path and straddling path. However, paths are removed under two conditions: (1) if there are two link disjoint paths, except the overlapping path. Moreover, there is not any destination node in the overlapping path; and (2) there is not any destination node in the straddling path. Results in [32] show that OPP-PRL can improve blocking probability about 10% over OPP-SDP when traffic load is high.

4.1.2. Source-Leaf Path Based Avoiding Residual Link (SLP-ARL) [32]

In SLP-ARL, a primary tree is first created for a given multicast session, and then the backup paths are found for the primary tree in order to avoid having residual links in the procedure of finding backup paths for a given working tree. Therefore, we can say that there are two stages in the SLP-ARL algorithm. In the first stage, a multicast routing algorithm is used to create a primary tree in order to reduce the number of leaf nodes on the multicast tree. Then, a backup path is derived for each destination node at the second stage. Hence, in this algorithm, backup paths are only setup for the leaf nodes on the primary tree. Note that the backup path and working path for a leaf node can overlap on some links, which have been

protected by other leaf nodes of the backup path. Therefore, by decreasing the number of leaf nodes on the primary tree, the number of backup paths and residual links go down. Simulation results in [32] indicate that SLR-ARL can improve blocking probability about 10% over OPP-SDP when traffic load is high.

4.2. Multicast Protection through Spanning Paths (MPSP) [33]

The MSPS protection scheme is a path-based protection scheme that derives a backup path for each spanning path and then selects part of these backup paths to protect the primary multicast tree. In Figure13a [33], there are *s* destinations (shaded nodes) in the multicast tree and *S* is the source node. In the multicast tree, there are three leaves, i.e., node *S*, node *C*, and node *E*. Thus, we can derive three spanning paths.

A Backup Path (BP) for a given spanning path can be derived as follows. The links along the given spanning path in the multicast tree and all the leaves close the two end nodes in the spanning path are removed, and then an auxiliary graph is considered. In order to protect the spanning path, the cost for each link is computed in the auxiliary graph. Finally, in the auxiliary graph, a least-cost backup path is obtained using a shortest path algorithm. Figure13.b to Figure13.d show the process of deriving backup paths from spanning paths. For example, to derive a backup path for the spanning path SP1= (S→A→B→C), links S→A, A→B, B→E, and the leaf C are deleted.

Now we can choose part of the backup paths to protect the primary multicast tree. Link Cost Gain of a Backup Path (LGBP) is defined as the ratio of the cost of a backup path over the cost of the spanning path corresponding to the backup path. For example, the LCGBP for SP1 in Figure13b [33] is 100/85 since the costs of SP1 and BP1 are 85 and 100, respectively. If LCGBP is a large number, it means that more spare capacity will be consumed to protect a given spanning path. With LCGBP, an auxiliary graph is created as follows. First, all leaves of the multicast tree are added into a graph. Second, for each node pair, if there exists a backup path for the spanning path between the node pair, add a link between them. Finally, assign the LCGBP of each backup path to its corresponding link.

Now, in order to choose the backup paths, the minimum spanning tree in the auxiliary graph should be obtained. The MST (Minimum Spanning Tree) algorithm [34] calculates the minimum spanning tree in the auxiliary graph. Derived backup paths for protecting the primary tree are shown in Figure13f [33].

After deriving the primary multicast tree and its backup paths, resources for the primary multicast tree and its protection paths are allocated as follows:

1) for each link in the set of links in the primary multicast tree, assign *w* units of bandwidth as working bandwidth; and
2) for each link in the set of links in the backup path, we should take intra-request sharing[4] into consideration in order to consume as small capacity as possible.

[4] One type of bandwidth sharing is intra-request sharing defined as follows. In local restoration, since each link on the primary path needs a backup path, and only one link failure can occur at any given time, the backup paths for different links in the same primary path can share links. Moreover, backup paths can share the primary path links.

The simulation results in [33] show that the MSPS scheme has better spare capacity utilization and blocking performances than the OPP-DSP scheme.

4.3. Shared Source-Leaf Path-Based Protection (SLPP) [35]

In the SLPP algorithm, the primary tree in spares-splitting WDM networks can share wavelengths with the backup paths without using any wavelength converters. In order to reduce implementation cost, nodes are divided into two types: (1) Multicast Capable (MC) that there are a small number of nodes in the network. MC is defined as a node which can forward an incoming message to multiple output channels; and (2) Multicast Incapable (MI) that cannot split the light.

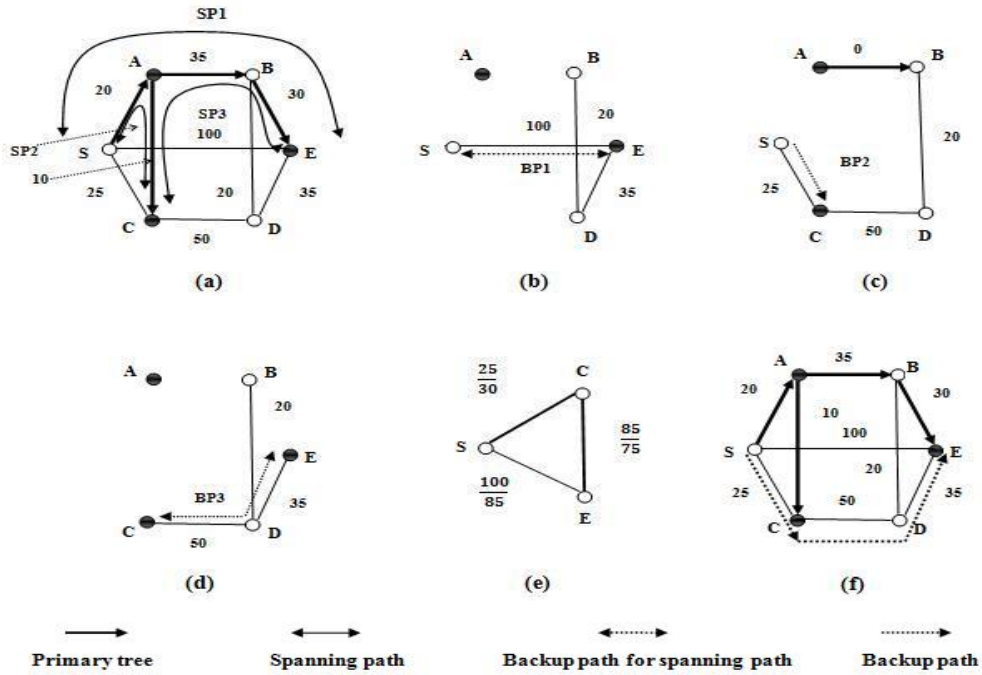


Figure 13. An overview of MPSP. In the figure, S is the source node, and shaded nodes are destinations. (a) illustration for spanning path; (b) deriving BP1;(c) deriving BP2; (d) deriving BP3; (e) selecting backup paths; (f) resulted backup paths.

First a primary tree with some MC and MI nodes are considered. For each leaf node on the primary tree, a link-disjoint backup path is discovered. Then all destinations on the primary path from the source to a certain leaf node are determined and the circles are formed by the primary and backup paths of the leaf node to protect these destination nodes. In order to share a wavelength, the auxiliary graph *G* should be created from the primary tree *P*. For this purpose, the SLPP algorithm is executed as follows:

The source node, the leaf nodes, and the MC nodes that are in *P* are located in *G*. Then, *X* is set equal to *G.* The costs of links in *X* are initially set to infinity. The links involved in *P* are removed from *X*. From each node to all other nodes in *X*, the shortest path is calculated. If

there is a shortest path between the pair of nodes, then an edge is added between them. Cost of the new edge is equal to the cost of the shortest path. Then, the backup path for each leaf node in X is found. For example, consider leaf node L of X. Primary path of source s to L is determined and then the cost of every link that is not in the primary path is set to infinity. The shortest backup path between s and L is found and these links are marked in G and P. The algorithm checks whether a marked link is located in both G and P. If a marked link exists in both G and P, then the cost of every link is set to zero and L is removed from the nodes of X. Finally, X is removed from the primary tree set. The algorithm is repeated for all trees, and backup paths are computed for all source and leaf nodes. In [35], SLPP has been compared with disjoint-segment (DS) and source-destination path pairs (SDPP). Simulation results show that the blocking probability of SLPP is lower than the blocking probability of both DS and SDPP.

4.4. Adaptive Shared Segment Protection (ASSP) [36]

The ASSP algorithm is a segment protection technique which does not require to identify the segments for a multicast tree in advance. ASSP is executed at two stages. A multicast tree is created firstly and then in order to protect paths of the multicast tree against link failures, backup paths are constructed. In this algorithm, s is the source node, D is the set of destination nodes, and N_t represents the set of nodes that are on the current multicast tree. Firstly, N_t is set to s.

To create multicast tree T, the first shortest path for every pair of nodes in N_t and D (u, v) is found. Every link in the shortest path is added to T. Every MC node on the shortest path is added to N_t if MC node is different from both u and v. Each MI node is removed from N_t if MI node = u and MI node $\neq s$. Finally, v is deleted from D and is added to the MI node.

After constructing the multicast tree, backup paths should be determined. Firstly, the connection degree of all nodes located on the tree are calculated. Moreover, the set of end nodes N_E on T is determined. The shortest path for any pair nodes in N_E on T is computed. Then, the backup path for every segment is calculated so that the calculated backup path should be the most efficient backup path among other backup paths. Note that each backup path should be link-disjointed from the corresponding segment. Backup paths are added to protecting paths set and the cost of each link on segments and backup paths is set to zero for resource sharing. It has been shown in [36] that the blocking probability of ASSP is lower than the link-disjoint algorithm and the shared disjoint path algorithm.

4.5. Shared Segment Protection with Re-Provisioning (SSPR) [37]

The SSPR is another proposed multicast protection scheme based on the shared segment protection. In this algorithm, a primary light tree is created and then a corresponding link-disjoint backup segment is requested for each multicast connection. Network nodes are equipped with MC-OXC, which have the full splitting capability. Moreover, each node has the full wavelength conversion capability. In this algorithm, it is assumed that only one connection request arrives at a time. In establishing dependable route stage, the multicast

connection is requested and then the working tree for the request is established. Working tree can be divided into some un-overlapped working segments and the backup segments are allocated to working segments. Costs of links are computed and then wavelength capacity is allocated/reserved for requested multicast connection. When a link fails in the network, the newly re-provisioned working segment and backup segment should be provided for working segments and for their corresponding backup segments that the failed link passes through them. For this purpose, the traffic of the primary working segments and backup segments should be restored and then their wavelength resources are released. Link-cost of each primary backup segment is computed and then a newly re-provisioned working segment with the minimum cost is computed. After allocating a wavelength to these segments, traffic is reverted from the primary backup segments to the newly re-provisioned working segment. For each failed primary backup segment, the least cost of the newly re-provisioned working segment is computed under link-disjoint constraint and then a new wavelength capacity is reserved for the newly re-provisioned backup segment. Finally, network state is updated. Note that both the MPH and the Dijkstra's algorithms are used in the SSPR algorithm in order to find minimum cost and shortest path parameters. The simulation results in [37] show that the blocking probability of SSPR is lower than the link shared protection and the path-pair shared protection.

4.6. Sparse Splitting Constrained Multicast Protection (SSMP) [38]

The wavelength channels can be shared with the primary tree in sparse splitting by SSMP algorithm and then the wavelength resources utilized by the multicast session are minimized. In this approach there are some definitions as follows:

- *Physical layer*: is used to map the network state and topology [38].
- *Physical edges*: each undirected link (x,y) in the physical network corresponding to physical edges(x_p,y_p) and (y_p, x_p) in the physical layer of an auxiliary graph (AG) [38].
- *Sharing layer*: is used to represent the primary wavelength channels that can be shared by backup paths [38].
- *Sharing edge*: It can be further divided into forward sharing (*FS*) edges and reversed sharing (*RS*) edges. For each edge (x,y) in a light tree, there is a FS edge from x_s to y_s in the sharing layer of AG. For each MC node (or leaf node) u on the light tree, there is an RS edge from node u_s to node v_s, where v_s is the parent MC node of node u on the light tree. The parent MC node of node u is defined as the nearest upstream MC node to u on the light tree [38].
- *Adding and dropping edges*: for each node x in the physical network, there is an adding edge from node x_p to x_s in auxiliary graph. For each MC node u on the light tree, there is a dropping edge from node u_s to u_p in the auxiliary graph [38].

The SSMP is applied to find the backup paths for each light forest, where a light forest is formed from multiple light trees with the sparse splitting and wavelength continuity constraints [38]. The SSMP algorithm is run in two phases:

1) *Generation of the auxiliary graph (AG):* In the generation phase, the map of each node x and corresponding physical edges of x are first added to the physical layer. Cost of physical edges (x_p,y_p) and (y_p, x_p) are set to infinity if the physical link (x,y) is used by the light tree. Costs of other physical edges in the physical layer are set to the corresponding link in the physical network. Moreover, the map of each node x and corresponding sharing edges of x are added to the sharing layer. The costs of the corresponding sharing edges are set to zero. Finally, the physical layer and sharing layer are connected together by adding and dropping edges. The costs of adding and dropping edges are set to zero.
2) *Finding the most efficient backup paths for leaf nodes:* In the second phase, backup path for each leaf should be found. For each link on the primary path from source node to leaf node l, the cost of corresponding FS edge (x_p, y_p) and *RS* edge starting at x_p in the sharing layer of *AG* are set to infinity. In order to determine the shortest path of (s_p, l_p) in the *AG*, the Dijkstra's algorithm is run. For each link over path of (x, y), the cost of corresponding *FS* edge (x_s, y_s) and *RS* edge starting at x_p in the sharing layer of the *AG* are set to zero. The leaf node u with minimum protection gain (see Eq.(9)) is selected. For each physical edge (x_p, y_p) belonging to backup path of (s_p,u_p), the cost of physical edges (x_p, y_p) and (y_p, x_p) in the physical layer are set to zero. Then, the corresponding physical link (x, y) is added to set B. This process is run for all leaf nodes. Finally, the light tree is removed from the primary light forest, if the primary light forest is equal to null; the B is shown as the backup topology in order to protect the primary light forest.

$$\text{Protection gain}=\frac{cost\ of\ the\ primary\ path\ to\ a\ leaf\ node}{cost\ of\ the\ backup\ path\ to\ a\ leaf\ node} \qquad (9)$$

4.7. Cross-Sharing [10]

In order to protect multicast sessions from any single link failure and to save cost of backup sources, a new cross-sharing mechanism has been introduced based on link-vector model in [10] which allows sharing of backup resources among several light-trees. This algorithm assumes that each link has a weight (c_{mn}) to represent the cost of moving traffic from one node to another. In addition, all nodes have multicast-capable opaque cross-connects to convert the signal from optical to electrical and then to optical domain.

In the first step, a minimum-cost primary tree is calculated for each multicast session j based on an Integer linear Program (ILP) [34]. In order to keep the topology of the network for multicast session j, matrix P^j is used. For each pair nodes m and n, P_{mn}^{j} is set to 1 if there is an edge from m to n; otherwise it is set to 0. Moreover, if this edge carries traffic to destination d_j, $P_{mn}^{jd_j}$ is also set to 1; otherwise it is set to 0. Then, for each d_j, a backup path $P_{mn}^{jd_j}$ is determined so that it is link-disjoint to $P_{mn}^{jd_j}$. In the second step, the link-vector of tree j is set for edge from m to n, $L_{mn}^{j\overline{mn}}$. Note that $\overline{m}$ and $\overline{n}$ are node indices where there is a failure between them. If an idle-backup edge of tree j on the edge from m to n becomes active for any destination d_j, $L_{mn}^{j\overline{mn}}$ is set to 1. Then, for every edge from m to n, the maximum required

number of idle-backup edges (v_{mn}) is determined. Finally, using an ILP and an appropriate solver (e.g, CPLEX [39]), the cost of idle-backup edges $\sum_{mn} c_{mn} \times v_{mn}$ is optimized.

4.8. Protection Algorithms under Reliability Constraints [40]

Most protection schemes provide 100% reliability that may not be necessary for some users. In other words, it is possible that some users only require lower connection reliability. The probability that a connection will work correctly in a period of time is called reliability and is determined based on environmental and man-made factors such as humidity, fires, and so on. The reliability of a multicast session is the least reliability of the paths from a source to relevant destinations. Three protection algorithms under reliability constraints have been proposed in [40] as follows, where Pruned Prim's Heuristic (PPH) [34] and Minimum-cost Path heuristic (MPH) [41] are used to establish a required minimum-cost Steiner tree [42]:

a. The first proposed algorithm is the ARC-disjoint tree with differentiated reliability (ADT_DiR). Arcs are unidirectional edges and arc-disjoint paths are paths that may share a link in the opposite direction. Moreover, two arc-disjoint trees may traverse a common link only in an opposite direction. ADT_DiR is run as follows:
   - A primary tree is created using either PPH or MPH.
   - The reliability of the primary tree is checked, if it is not smaller than the user's requirement, the multicast request is accepted; otherwise the arcs along the primary tree is removed and a backup tree is constructed in the remaining graph using either PPH or MPH.

b. Partial tree protection (PTP) is the second protection algorithm suggested in [40]:
   - A primary tree is created using either PPH or MPH.
   - The reliability of the primary tree is checked. If it is not smaller than the user's requirement, the multicast request is accepted and waits for another multicast request. Otherwise, to create a small tree, unwanted branches of the primary tree are pruned on which the destination satisfies the user's requirement without any effect on other destinations.
   - An arc-disjoint tree is built for the small tree.

c. The third proposed algorithm is Segment Protection with Most-likely Failure Link (SP_MFL). Note that MFL means the fiber link whose reliable probability is the lowest in the path. SP_MFL is executed as follows:
   - A primary tree is created using either PPH or MPH.
   - The reliability of the primary tree is checked, if it is not smaller than the user's requirement, the multicast request is accepted and waits for another multicast request; otherwise, go to the next step.
   - The primary segments on the primary tree are identified.
   - The set T is created, where it involves the paths that have lower reliability than required reliability.
   - The least reliability path of T is selected and divided into several segments.
   - In order to meet the required reliability, the segments are chosen to be protected by the segment protection algorithm (see Section 2.1.5).

- The reliability of all paths of T should be calculated. If the T is empty, the multicast request is accepted.

The simulation results show that these three algorithms decrease the blocking ratio and consumed protection resources.

4.9. A Tree Protection Scheme with Optimized Shortest Path Tree (TP-OSPT) [9]

According to the heterogeneous nature of traffic in WDM, the TP-OSPT has been suggested as a new protection scheme to support unicast and multicast connection requests. For a multicast request with source *s* under the tree protection scheme with shortest path tree routing heuristic (TP-SPT), the shortest paths from *s* to all the multicast destinations are found using the Dijkstra algorithm and then the combination of all the shortest paths will construct a multicast tree. However, sometimes it is impossible to find a link-disjoint tree from a source to the multicast destination using TP-SPT [9]. Therefore, in order to increase the sharing in the multicast tree, the TP-OSPT has been suggested.

In TP-OSPT, traffic is assumed to be a combination of unicast and multicast connections and each node is equipped with a multicast-capable all optical switch. The TP-OSPT algorithm is executed for each multicast destination of the multicast session with source *s*. At first, the minimum-cost path from source *s* to its multicast destination is computed by the Dijkstra algorithm. Then, the cost for all links included in the minimum-cost path is set to zero. The combination of all the minimum cost paths will construct a multicast tree. Simulation results show that TP-OSPT has a good performance results in moderate and large group sizes.

5. PERFORMANCE COMPARISION

To insight on suitable protection schemes, in this section, we compare the protection schemes with each other.

5.1. Comparison among Basic Traffic Protection Techniques

The following are some comparison comments among the traffic protection schemes that have been explained in Section 3:

- Path-protection can provide many protection paths, so that the possibility of not finding a path is very small. In the Link Protection, however, it is hard to find such paths for all links.
- Path Protection can protect even against failures that cannot be protected by Link Protection like signal degradation.

- Path Protection spends longer time for service restoration compared to Link-based Shared protection.
- Path-based Shared protection can immensely save resource protection capacity as compared with Link-based Shared protection and Path-based Dedicated Protection. However, the restoration time is longer and increases computations.
- Using p-cycle is much more capacity efficient than using dedicated path protection.
- Link-based Shared protection spends longer time for service restoration compared to Link-based Dedicated Protection.
- Link-based Dedicated Protection has more reserved capacity than Link-based Shared protection.
- Usually Virtual Link-based Shared Protection has a less capacity comparing to Virtual Link-based Dedicated Protection.
- Virtual Link-based Shared Protection has longer restoration time compared to Virtual Link-based Dedicated Protection.
- Segment Protection has a faster recovery in proportion to Path-based Shared Protection because the backup segments usually are shorter than backup paths used in Path Protection.
- The 1:N protection is more efficient in terms of protection resources as compared to 1+1 protection, but it still needs time to detect failure and to reroute signals.
- The 1+N is fast and inexpensive recovery from failures by providing two copies of the same signal on two disjoint paths.

5.2. Comparison among Protection Techniques Supporting Unicast Traffic

In this section, we compare the complexity and blocking probability of protection schemes that can support unicast traffic with each other. For all illustrated algorithms, the information of blocking probabilities has been obtained from their relevant references ([14,22-30]). Table 3 shows the complexity of the algorithms, where V is the set of network nodes, and D is the set of destination nodes. Moreover, K is the number of feasible cycles to every possible destination, T is the time to derive a backup segment, N is the number of network nodes, W is the number of wavelengths, L is the set of links, and n is the number of nodes of the working paths. Moreover, K is the number of cycle in the FT-ABR algorithm and k is the number of candidate working and candidate backup paths. In the following, when information is not available for a parameter, we use term Not Available (*NA*).

According to Table 4, we can see that for W=16, TS has a lower blocking probability than RSSP at high load, but blocking probability of RSSP is low at light load. However, TS has lower complexity than RSSP (see Table 3). Moreover, at W=8, LSA has the lower blocking probability than FT-ABR and HCSPP.

5.3. Comparison among Protection Techniques Supporting Multicast Traffic

In this section, we compare the protection schemes that support multicast traffic with each other. In multicast protection schemes, the complexity and blocking probability are important parameters. For all illustrated algorithms, the information of blocking probabilities

has been obtained from their relevant references (9, [32-40]). All algorithms are evaluated by simulation implemented for USnet. Table 5 shows the complexity of the algorithms, where V is the set of network nodes, D is the set of destination nodes, and S represents the number of MC nodes. Moreover, the number of un-overlapped working segments on a primary light-tree is showed with m.

Table 3. Complexity of unicast protection algorithms

Protection algorithm	Complexity
Non-SRLG	NA
Pd-SPP	$O(kN^3)+kO(N^2)$
TS	$O(NT)$
HCSPP	$(2 + D^2)O(N^2) + O(D^2)$.
OHS	NA
SPAVT	$O(\|L\|+\|L\|.\log\|N\|)$ & $O(\|k^2.\|N\|^8\|)$
FT-ABR	$O(KN^2W)$
RSSP	$(n(n-1))O(N^2)$
Network coding	NA
LSA	NA

Table 4. Blocking probability (%) vs. traffic load (Erlang) for unicast protection algorithms

Protection Algorithm	*W*	*N*	Blocking probability (%) vs. traffic load (Erlang)				
			30	40	60	80	100
Non-SRLG	NA	NA	NA	NA	NA	NA	NA
Pd-SPP	16	21	14	19	31	40.5	50.4
TS	16	16	0	0	5	11	20
HCSPP	8	21	2.53	2.642	11.123	23.542	32.365
OHS	4	12	15.2	18.5	22.3	26.5	34.5
SPAVT	NA	NA	NA	NA	NA	NA	NA
FT-ABR	8	14	0.5	1	5.2	8.4	10.2
RSSP	16	16	-----	0.55	1	10	35
Network coding	NA	NA	NA	NA	NA	NA	NA
LSA	8	24	0.1	1.5	2.5	4	5

Table 5. Complexity of multicast protection algorithms

Protection algorithm	Complexity
OPP-PRL	$O(\|D\|\|V\|^2)$
SLP-ARL	$O(\|D\|\|V\|^2)$
MPSP	$O(\|V\|^2\|D\|^2)$
SLPP	$O(\|S\|\|D\|\|V\|^2)$
ASSP	$O(\|V\|^5)$
SSPR	$O(\|V\|\|D\|^2+m\|V\|^2)$
SSMP	$O(2(\|D\|+\|D\|^2).\|V\|^2)$

Table 5 shows that OPP-PRL and SLP-ARL have the same complexity, but we can see that SLP-ARL has lower blocking percentage than OPP-PRL when traffic load is high (see Figure14) since the SLP-ARL algorithm uses the links with least number of wavelengths [32]. Moreover, Figure14 shows that blocking probability of MPSP is lower than OPP-PRL and SLR-ARL because average cost of MPSP is smaller, and therefore, MPSP is able to share spare capacity when backup path is derived. However, the complexity of MPSP is higher than OPP-PRL and SLR-ARL.

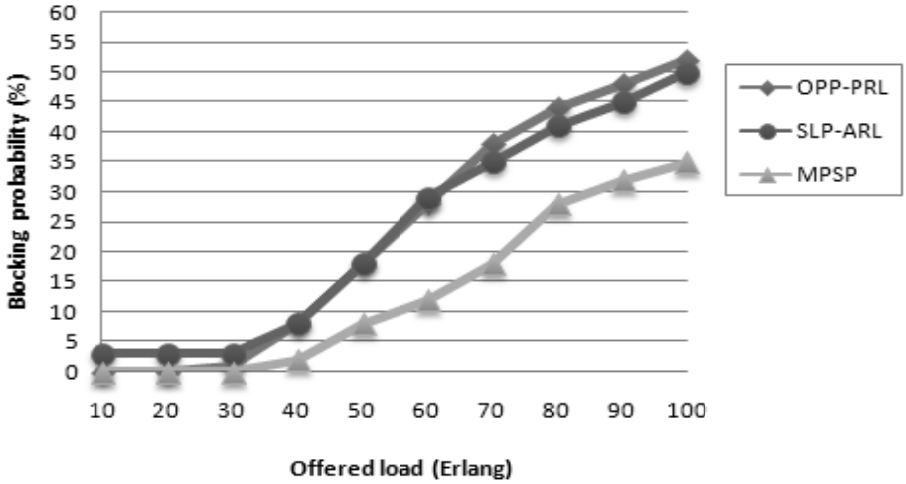


Figure 14. Blocking probability vs. traffic load when the number of wavelengths and session size are set to 32 and 10, respectively.

If a multicast tree has some circles, it can result in better performance [36]. This is why ASSP uses some circles to enhance its performance. Moreover, ASSP does not need to identify segments in advance. Therefore, ASSP can reduce blocking probability by increasing *W* (see Table 6). However, Table 6 shows that the complexity of ASSP is higher than other algorithms.

Table 6. Blocking probability vs. traffic load (Erlang)

Algorithm	*N*	*W*	Traffic load (Erlang)							
			30	40	50	60	70	80	90	100
ASSP	10	16	22	37	46	54	62	65	68	71
		32	0.5	3	7.2	16.9	26	32	40	44.8
		64	0	0	0	0	0.5	0.8	2.6	4.9
SSPR	15	64	0	0.8	1	3	7.5	14.8	30	42

Table 7. Blocking probability vs. session size

Algorithm	Traffic load (Erlang)	*W*	Session size						
			2	6	10	12	14	16	18
SLPP	80	64	0.2	19.2	25	26.1	28	29.6	30.9
MPSP	100	64	0	2	4.8	7	13.2	17.1	19.6

The SSPR algorithm has very small blocking probability at low traffic loads (see Table 7) because SSPR can afford more flexible backup capacity sharing. However, the complexity of SSPR grows up if the number of un-overlapped working segments on a primary light-tree increase. Under the SLPP algorithm, the network nodes do not use any wavelength converters in order to reduce the cost of implementations. However, SLPP has higher complexity than OPP-PRL and SLP-ARL. Moreover, with respect to Table 8, one can see that the blocking probability of SLPP is higher than MPSP when traffic load of SLPP is lower than MPSP's traffic load. This is because the MPSP algorithm can share spare capacity when backup path is obtained. With respect to Figure15, SSMP performs better than MPSP for all traffic loads. This is because SSMP consumes smaller average network cost than MPSP [38]. The two algorithms have the same computation complexity in theory.

The simulation results (see Tables 8 and 9) show that CSP has the lowest blocking ratio and the best performance. In addition, ADT_DiR has a higher blocking ratio than PTP because it consumes more resources to meet user's requirements [40]. However, both ADT_DiR and PTP have better performance results than ADT.

According to Table 10, TP-OSPT has smaller blocking ration than TP-SPT in all multicast group sizes. Note that multicast group size is defined as the maximum percentage of the network nodes that could be destination nodes [5].

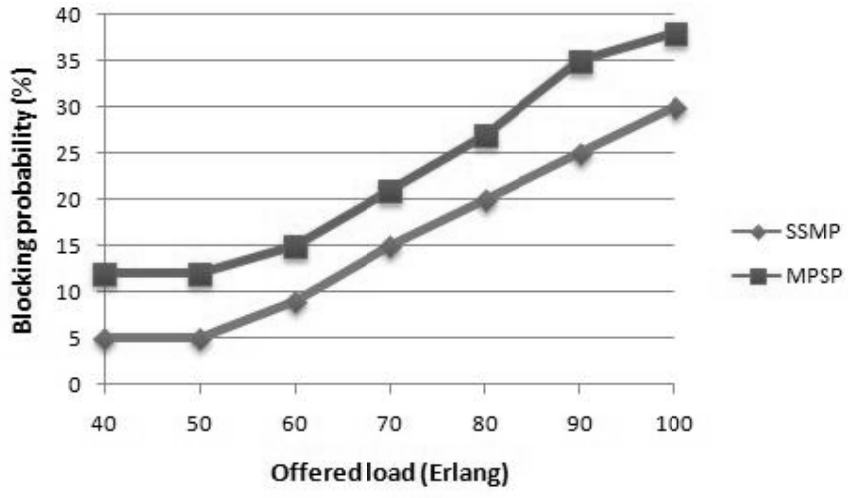


Figure 15. Comparison of blocking probability between MPSP and SSMP when the number of wavelengths is 64 and session size is 10.

Table 8. Blocking probability vs. session size when W=32, load is 80 Erlangs and required reliability is 0.90

Algorithm	Session size						
	3	6	9	12	15	18	21
ADT	0.1	0.12	0.2	0.3	0.345	0.355	0.37
ADT_DiR	0.01	0.02	0.05	0.11	0.21	0.24	0.265
CSP	0.01	0.02	0.03	0.05	0.06	0.08	0.1
PTP	0.01	0.02	0.04	0.08	0.12	0.14	0.16

Table 9. Blocking probability vs. session size when W=32, load is 80 Erlangs and required reliability is 0.96

Algorithm	Session size						
	3	6	9	12	15	18	21
ADT	0.1	0.12	0.14	0.29	0.345	0.355	0.37
ADT_DiR	0.035	0.05	0.15	0.26	0.33	0.34	0.355
CSP	0.01	0.04	0.11	0.17	0.18	0.2	0.21
PTP	0.35	0.05	0.13	0.18	0.235	0.24	0.26

Table 10. Blocking probability vs. multicast group size when W=64, and load is 10 Erlangs

Algorithm	Multicast group size									
	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
TP-SPT	0	0.04	0.14	0.019	0.25	0.275	0.29	0.335	0.345	0.35
TP-OSPT	0	0.02	0.035	0.04	0.045	0.045	0.048	0.05	0.05	0.05

CONCLUSION

Network survivability is an important concern for WDM optical networks. In this chapter, we have reviewed the protection schemes that are used to survive link failures in optical networks supporting the unicast and multicast traffic. According to our study, we have found out that the best schemes for unicast traffic could be the techniques that are based on the dedicated protection, if the capacity is not of our main concern because these schemes have low recovery time and network complexity. Otherwise, when either the network complexity is not important, or the network is not sensitive to the recovery time, the schemes that are based on the shared protection and the schemes that divide networks into several areas and *p*-cycle are better choices. Note that the path-based shared protection needs more capacity compared to other schemes based on the shared protection. Among the schemes that are based on the shared protection or sub protection, the schemes that protect a whole primary path have longer recovery time compared to the schemes that protect either a link or a segment in a primary path.

In unicast protection algorithm, we saw that for the same number of wavelengths, TS has a lower blocking probability than RSSP in higher load, but blocking probability of RSSP is lower at light loads. However, TS has lower complexity than RSSP. Moreover, for the same number of wavelengths, LSA has the lower blocking capability than FT-ABR and HCSPP.

The basic protection schemes have some challenges in tree networks. For example, 1+1 and 1:1 schemes include excessive use of resources and are unable to discover link disjoint trees in a mesh network, which may lead to the blocking of a large number of multicast sessions while trying to establish them. In addition, in path protection, a backup link disjoint path in the multicast tree from the source to a given destination node is derived. Consequently, this scheme does not consider spare capacity sharing among different multicast trees. Moreover, any change in cycle network topology or demand will need every node in the network to re-compute the protection/restoration cycle path, even though the change or failure

might have occurred in a far distant from it. However, network trees provide more flexibility and scalability. Therefore, other algorithms appropriate for tree networks have been designed. For the protection of multicast traffic, according to our study, we have found out that if the complexity is not of our main concern in multicast protection, SSMP is the best technique for a high number of wavelengths, because it has lower blocking probability. Otherwise, when the network complexity is important, the OPP_PRL and SLP_ARL techniques are better choices. Note that if the number of un-overlapped working segments on a primary light-tree is low, SSPR can be selected when complexity is important.

REFERENCES

[1] Hülsermann. R., Jäger, M., Koster, C. A., Orlowski, S., Wessäly, R., $ Zymolka, A. (2006) Availability and Cost Based Evaluation of Demand-wise Shared Protection. *Proc. 7th ITG-Workshop on Photonic Networks,* VDE-Verl., 161-168.

[2] Hyytia, E. (Feb.2001). Dynamic Control of All-Optical WDM Networks. *Licentiate thesis, Helsinki University of Technology, Networking Laboratory*.

[3] Kiaei, M., & CH.Assi, (2009). A survey on the p-cycle protection method. *IEEE Communications Surveys and Tutorials*, 11(3), 53-70.

[4] Asthana, R. (*Nov*.2007). Study of p-cycle based protection in optical networks. *Doctor of Philosophy thesis, Department of electrical engineering of Indian Institute of technology, kanpur, India.*

[5] Maier, G., & Pattavina, A. (2002). Optical Network Survivability: Protection Techniques in the WDM Layer. *Photonic Network Communications*, 4:3/4, 251-269.

[6] Ho, P.H., & T. Mouftah,H. (2004). Shared protection in mesh WDM networks. *IEEE Communications Magazine.*

[7] Vinodkrishnan K., Chandhok N., Durresi A., Jain R., Jagannathan R., and Seetharaman S., (2001) Survivability in IP over WDM networks. *Journal of High Speed Networks*, 10(2), 79-90.

[8] Daeinabi, A., Ghaffar pour Rahbar, A., & Mardini, W. (2008). Protection of Multicast Traffic in WDM Mesh Optical Networks," *Proc. IEEE 5th International Symposium on High Capacity Optical Networks and Enabling Technologies*, 192-197.

[9] Khalil, A., Hadjiantonis, A., Ellinas, G., & Ali, M. (2006). Dynamic provisioning of survivable heterogeneous multicast and unicast traffic in WDM networks. *IEEE ICC*, 2465-2470.

[10] Singhal, N., Ou, C., & Mukherjee, B. (2006). Cross-sharing vs. self-sharing trees for protecting multicast sessions in mesh networks. *Computer Networks*, 50(2), 200-206.

[11] Wang, H., Modiano, E., & Medard, M. (2002). Partial Path Protection for WDM Network: End-to-End Recovery Using Local Failure Information. *Proc. 7th International Symposium on Computers and Communications (ISCC2002)*, 719-725.

[12] Xin, Y., & Rouskas, G.N.(jan.2004). A Study of Path Protection in Large-Scale Optical Network. *Photonic Network Communications,* 7 (3), 267-278.

[13] Tiyachate, R. Protection in Survivable WDM Grooming Network, *Master thesis*, *Department of Electrical and Computer Engineering, NC State University*, 2004-02-05.

[14] Stidsen, T., & Thomodsen, T. (May.2005). Joint Routing and Protection Using *p*-cycle. *Technical Report, Technical University of Denmark, Informatics and Mathematical Modeling*.

[15] Luo, H., Li, L., & Yu, H. (2007). Insights for segment protection in survivable WDM mesh networks with SRLG constraints. *Photon Network Communications,* 14, 361-368.

[16] Schuplce, D.A. (July.2006). Analysis of p-cycle Capacity in WDM Network. *Photonic Network Communication, Springer,* 12(1), 41-51.

[17] Xu, D., Xiong, Y., & Qiao, CH. (Oct.2003). Novel Algorithms for Shared Segment Protection. *IEEE Journal on Selected Areas in Communication*, 21(8), 1320-1331.

[18] Yuan, SH., & Jue, J.P. (Aug.2001). Shared Protection Routing Algorithm for Optical Network. *Proc. OptiComm 2001 Conference*, Denver, USA.

[19] Ramasubramanian, S. (Nov.2004). Supporting multiple Protection Strategies in Optical Networks. *Technical Report*, *Department of Electrical and Computer Engineering, University of Arizona*.

[20] Xue, G., Zhang, W., & Thulasiraman, K. (2005). Establishment of Survivable Connections in WDM Networks using Partial Path Protection. *Proc. IEEE ICC'* 05, 1756-1760.

[21] Ou, C., Zang, H., & Mukherjee, B. (Nov.2004). Protection for Scalability and Fast Recovery in WDM Mesh Networks. *IEEE Journal on Selected Areas in Communication*, 22, 1859-1875.

[22] Cao, J., Guo, L., Yu, H., & Li, L. (2006). A novel hop constrained sub-path protection routing algorithm in WDM networks. *Optics Communications,* 260, 155-163.

[23] Cao, J., Guo, L., Yu, H., & Li, L. (2007). Partial SRLG-disjoint shared path protection with differentiated reliability in survivable WDM network. *Int. J. Electron. Commun. (AEU),* 61, 353-362.

[24] Bicudo, M., Moraes, L., Laufer, R., cunha, D., Velloso, P., & Duarte, O. (May.2005). Protection and Minimal Interference in WDM Mesh Network. *Proc. ICT' 05*, South Africa.

[25] Zhang, X., Li, L., Wang, SH., & Liao, D. (2009). OHS: A Novel hybrid survivability approach for WDM mesh networks. *Int. J. Electron. Commun.* (*AEÜ*), 63(8), 708-711.

[26] Wang, X., Guo, L., Wang, X., Zhang, Y., Zheng, X., Huo, W., Li, H., & Wang, H. (2009). A new algorithm based on auxiliary virtual topology for sub-path protection in WDM optical networks. *Computer Communications*, 32(4), 777-781.

[27] Ngo, S.H., Jiang, X., & Le, V.T. (2006). Ant-based survivable routing in dynamic WDM networks with shared backup paths. *J Supercomputing*, 36, 297-307.

[28] Cao, J., Guo, L., Yu, H., & Li, L. (2007). A novel recursive shared segment protection algorithm in survivable WDM networks. *Journal of Network and Computer Applications,* 30, 677–694.

[29] Kamal, A. E. (Nov.2007). Using Network Coding over p-Cycles for Survivable Network Operation. *Proc. IEEE 4th International Symposium on High Capacity Optical Networks and Enabling Technologies (HONET 2007),* Dubai.

[30] Wei, X., Li, L., Yu, H., & Guo, L. (2007). A Heuristic Algorithm for Priority-based lightpath Allocation in Survivable WDM Mesh Networks. *Proc. of SPIE*, vol.6784.

[31] Shah Heydari, S., & Yang, O. (2001). A Tree-based Algorithm for Protection/Restoration in Optical Mesh Networks. *Electrical and Computer Engineering*, Canadian conference, 2, 1169-1174.

[32] Wang, X., Wang, S., Li, L., & Song, Y. (2007). Dynamic Multicast Protection Algorithms for Reducing Residual Links in WDM Mesh Network. *Computer and Communications, ICCC 2007*, 275-281.

[33] Luo, H., Yu, H., Li, L., & Wang, SH. (June.2006). On Protecting Dynamic Multicast Sessions in Survivable Mesh WDM Networks. *Proc. ICC' 06*, 2, 835 - 840.

[34] Cormen, T., Leiserson, C., Rivest, R., & Stein, C. *Introduction to Algorithms*, 2nd Edition, MIT Press and McGraw-Hill, 2001.

[35] Wang, X., Wang, SH., & Li, L. (2007). Protection Multicast Sessions in WDM Networks with Sparse-Splitting Constraints. *ETRI Journal*, 29(4), 524-526.

[36] Lu, C., Luo, H., Wang, SH., & Li, L. (2006). A novel Shared Segment Protection Algorithm for Multicast Sessions in Mesh WDM Networks. *ETRI Journal*, 28(.3), 329-336.

[37] Liao, L., Li, L., & Wang, SH. (2008). Multicast Protection Scheme in Survivable WDM Optical Networks. *Journal of Network and Computer Applications,* 31, 303-316.

[38] Wang, X., Wang, SH., & Li, L. (2009). Protecting light forest in survivable WDM mesh networks with sparse light splitting. *Int. J. Electron. Commun.* (*AEÜ*), 63(12), 1043-1053.

[39] http://www-01.ibm.com/software/integration/optimization/cplex/

[40] Lin, R., Wang, S., & Li, L. (2007). Protections for multicast session in WDM optical networks under reliability constraints. *Journal of Network and Computer Applications*, 30(2), 695-705.

[41] Takahashi, H., & Matsuyama, A. (1980). An approximate solution for the Steiner problem in graphs," *Math Jpn*, 573–577.

[42] Hakimi, SL. (1971). Steiner's problem in graphs and its implications," *Networks*, 1(2), 113–133.

[43] Singhal, N., & Mukherjee, B. (April.2003). Protecting Multicast Sessions in WDM Optical Mesh Networks. *Journal of Lightwave Technology*, 21 (4), 884-892.

In: Computer Science Research and Technology. Volume 2 ISBN 978-1-61122-073-5
Editor: Janice P. Bauer

Chapter 2

CLOUD COMPUTING: MAKING HEADWAY WITH THE SCIENTIFIC COMMUNITY

Nabil Sultan
Faculty of Business and Computer Science, Liverpool Hope University,
United Kingdom

ABSTRACT

Cloud computing is an emerging new computing paradigm designed to deliver numerous computing services in ways that have not been experienced before. It is increasingly being viewed as a utility-oriented computing service. There are even speculations that it has the potential to be the 5th utility (after water, electricity, gas and telephony). This computing service is underpinned by a number of existing technologies such as the Internet, virtualization, grid computing and Web services. The flexibility of this computing service approach and its provision in a pay-as-you-go way have opened many possibilities for organizations, of all types, that did not exist before. Among those organizations are those engaged in scientific research. A number of cloud computing services are now emerging to cater for the scientific community, with a potential to bring substantial cost-saving and efficiency advantages. This chapter will highlight those issues and also analyze some of the limitations that are currently associated with this computing service modality.

Keywords: Cloud Computing, Web Services, Virtualization, Virtual Machines (VMs), Grid Computing, High Performance Computing (HPC), Small and Medium Enterprises (SMEs).

I. INTRODUCTION

The provision of software as a service (SaaS) is not a new computing practice. In fact, it predates the emergence of computers themselves. In the 1930s some companies (such as IBM) specialized in producing electric accounting machines based on punch-cards and were

able to offer data processing services (e.g., payrolls) to organizations. Providers of such services operated many 'service bureaus' where customers would bring their data for processing in return for a fee. Organizations that were unable to purchase those data processing equipment found it economically viable to pay for those services. Then came mainframe computers in the 1950s and 1960s which continued this practice that became known as 'timesharing'. Organizations that were unable to afford buying mainframes computers would rent the data processing functionality of those machines from a number of providers. Connection to mainframes was achieved through a normal telephone line connecting those massive machines and 'teletype', replaced afterwards with better visual display machines, at the clients' end. Then came personal computers which effectively killed timesharing due to the affordability of personal computers and also the flexibility of the software used on them. Campbell-Kelly (2009) argues that the very things that killed the timesharing industry in the 1980s have reversed in that computing infrastructure has become increasingly complex and expensive to maintain due, for example, to issues relating to security and constant software upgrades thus making cloud computing a more economically viable alternative. One analyst called cloud computing 'timesharing 2.0' in reference to the old practice of buying computing resources on demand (Campbell, 2009).

However, despite the popularity of personal computers, a form of utility computing still existed. Many software providers, known as Application Service Providers (ASPs), emerged in the 1990s to provide organizations with software as a service via the medium, this time, of the Internet. These early attempts at "utility computing" did not prove to be popular. There were two main reasons for this. First, most of the software offered by those providers was proprietary, which meant organizations using this type of service cannot change providers very easily, i.e., they were vendor-locked. Second, lack of sufficient bandwidth was another problem. During the 1990s broadband was neither cheap nor plentiful enough to deliver computing services with the required speed and reliability (Carr, 2009). Then came Web services (especially those based on the XML-based SOAP message protocol) that promised to deliver highly portable software remotely, through the medium of the Internet, without any ties to any platform (i.e., operating systems) or programming languages. Web services heralded a new era of 'software as a service' (SaaS).

The idea of delivering software remotely took a new turn with the emergence of cloud computing. Not only can software be consumed remotely, it can also be consumed as and when needed through a pay-as-you-go cost structure. Cloud computing also promises many other exciting possibilities where, not only software but also other computing-related functionality can be consumed remotely as and when needed, thanks to other relatively new technologies such as virtualization and grid computing.

Virtualization is a technology that masks the physical characteristics of computing resources (e.g., a PC, a Server) in order to simplify the way in which other systems, applications, or end users interact with them. For example, a PC running Windows can use virtualization to enable another operating system (e.g., Linux) to run besides Windows. Furthermore, the technology also enables single physical resources (e.g., a server, an operating system, an application, or storage device) appear as multiple logical resources.

Grid computing involves the use of software to combine the computational power of many computers, connected in a grid, in order to solve a single problem, often one that requires a great deal of computer processing power. Furthermore, grid computing also uses software that can divide and farm out pieces of a program to as many as several thousand

computers. Grid technology, therefore, can be thought of as the technology that enables the establishment of network-distributed parallel processing and distributed and large-scale cluster computing.

Both virtualization and grid computing technologies have become two fundamental building blocs of cloud computing.

II. CLOUD COMPUTING: DEFINITION

A Web search for cloud computing is likely to reveal a stream of definitions. There seems to be no common standard or definition for cloud computing (Voas and Zhang, 2009; Grossman, 2009). A study conducted by McKinsey (the global management consulting firm) found 22 definitions for cloud computing. A more commonly used definition describes it as clusters of distributed computers (largely vast data centers and server farms) which provide on-demand resources and services over a networked medium (usually the Internet). The term "cloud" was probably inspired by cloud illustrations depicting the Internet which are often found in many computing textbooks and journals.

In order to understand what cloud computing is all about it is perhaps useful to have an idea of the type of services it offers for consumers. The following is a list of the three main types of services offered by cloud computing (Sultan, 2009):

- Infrastructure as a Service (IaaS): Products offered via this mode include the remote delivery (through the Internet) of a full computer infrastructure (e.g., virtual computers, servers, storage devices etc.);
- Platform as a Service (PaaS): To understand this cloud computing layer, one needs to remember the traditional computing model where each application managed locally required hardware, an operating system, a database, middleware, Web servers, and other software. One also needs to remember the team of network, database and system management experts that are needed to keep everything up and running. With cloud computing, these services are now provided remotely by cloud providers under this layer;
- Software as a Service (Saas): Under this layer, applications are delivered through the medium of the Internet as a service. Instead of installing and maintaining software, you simply access it via the Internet, freeing yourself from complex software and hardware management. This type of cloud service offers a complete application functionality that ranges from productivity (e.g., office-type) applications to programs such as those for Customer Relationship Management (CRM) or Enterprise Resource Management (ERM).

Despite the flexibility of cloud computing, it should not be assumed that cloud products offered by any of the above services are likely to work out-of-the-box (i.e., they are not purely plug-and-play). In some cases they might. Messaging and collaboration cloud products such as Google Apps are perhaps examples of those out-of-the-box products, even though they still require some level of configuration nevertheless. Some degree of development (i.e.,

programming) will be required through the use of the cloud providers' APIs (application programming interfaces). These are the programming instructions created and offered by the cloud service providers to those who want to access the functionality of their products.

Currently, some of those APIs are proprietary. This is an issue which will be explored later when examining some of the limitations and concerns of cloud computing.

However, the notion of providing a wide array of computing-related services on the fly on a pay-as-you-go basis opens many opportunities for the providers of those services to exploit this expanding market which (according to Merrill Lynch) is worth 100 billion US dollars (Buyya, Yeo and Venugopal, 2009). At the same time, it increases the options available to policy makers entrusted with the job of ensuring the efficient functioning of their organization's IT resources. On that basis, cloud computing probably represents a paradigm shift in the way IT, in its all aspects, is being viewed by commoditising it in a manner that was not possible before (Sultan, 2010).

III. PUBLIC, PRIVATE AND HYBRID CLOUDS

Cloud services, such as those listed above, are known as public clouds when offered by commercial cloud providers. This is to distinguish them from private clouds, also known as internal or corporate clouds, where the cloud infrastructure (e.g. data centers, servers etc.) and services are provided, housed and maintained internally by the users themselves. This is an example of organizations becoming themselves the providers and consumers of cloud services. This approach, which might provide organizations with total security and control of their own resources, actually defeats the whole utility-oriented purpose of cloud computing which is supposed to guarantee choice, flexibility and cost-effectiveness. In fact, it is doubtful if this type of arrangement has anything to do with cloud computing as a concept which is largely utility-based. A number of public cloud providers are now coming up with solutions which incorporate elements of both private and public clouds that might be classified as "hybrid clouds". For example, companies such as Amazon have very recently launched a service known as Virtual Private Cloud (VPC) designed to address this issue of security and control which seems to be one of the concerns of large organizations. Amazon's VPC allows organizations to connect their existing legacy infrastructures to Amazon's clouds via a Virtual Private Network (VPN) connection [1]. This means that Amazon's cloud customers can create their own Amazon isolated resources, i.e., virtual private clouds (VPCs) within Amazon's cloud infrastructure and then connect those resources directly to their network servers using industry-standard encrypted IPsec VPN connections. In doing so, these customers will be able to extend their existing security and management policies within their IT infrastructure to their VPCs as if they were running within their own infrastructures.

[1] A virtual private network (VPN) is a private network that provides remote access to its resources through a public telecommunication infrastructure, such as the Internet.

IV. Public Cloud Services for HPC

There are many public cloud computing services on offer. Many of those serve a variety of tasks such as enabling users to: create hundreds, even thousands, of virtual machines (e.g., virtual servers) at the click of button; acquire unlimited space to save their data; access the required amount of software functionality for office-related tasks (e.g., word processing, spreadsheets etc.) and business-related operations (e.g., CRM). If fact, there is now a potential to create one's entire IT infrastructure in the cloud. A Californian startup company by the name of 3Tera has developed software, named AppLogic, that can automate the creation of complex corporate systems. The software provides a simple graphical user interface which a designer can use to drag and drop icons representing traditional components (e.g., databases, routers, firewalls, cables etc.) onto a page in a Web browser. When the designer has the arrangement in place, he or she can then click a button and the software creates the virtual IT infrastructure in a matter of minutes (Carr, 2009).

Cloud computing for high performance computing (HPC) has not been on the priority list of many cloud providers. However, this attitude is now changing as cloud providers have begun to realize the business potential of this area of computing. Currently, there are few cloud providers that can declare themselves as having cloud solutions for HPC. However, two cloud providers, namely Amazon and Microsoft, have two cloud platforms that have the potential to provide the high performance computing required by the scientific community. The first product is Elastic Compute Cloud (EC2) from Amazon and the second is Azure from Microsoft. Amazon's EC2 is part of Amazon Web Services (AWS). It offers a simple web service interface that allows users to request, monitor, and manage any number of VM instances running on physical compute nodes in their datacenters. The system therefore enables users to scale capacity, both up and down, as their computing requirements change. Compute nodes are organized into CPU capacity (several CPUs), RAM, and local storage. The user can programmatically start and stop compute node instances to deal with increasing or decreasing demand. EC2 is now increasingly been targeted by the scientific community for HPC purposes and is proving to be useful (see The MathWorks, 2008; Rehr et al, 2008).

Azure is a complex beast. It has three components: Windows Azure, SQL Azure and Azure AppFabric (formerly known as .Net Services). Windows Azure is designed to provide developers with on-demand compute and storage to host, scale, and manage Internet or cloud applications. SQL Azure is developed to extend the capabilities of SQL Server, Microsoft's database managing system (DBMS), into the cloud as a Web-based distributed relational database. AppFabric is a set of integrated technologies designed to help developers connect applications and services between Windows Azure and on-premises deployments (see Foley, 2009; Bailey, 2009). Microsoft is devoting increasing sources behind this platform such as adding more data centers. This has made Azure attractive for some HPC projects as demonstrated by some of Microsoft's case studies (Microsoft, 2009a).

An increasing number of small HPC providers are now migrating some of their services to the cloud. For example, SGI, an HPC solution provider, recently launched a cloud-based platform named 'Cyclone'. Cyclone is expected to, initially, support five technical domains: computational fluid dynamics, finite element analysis, computational chemistry and materials, computational biology, and ontologies (Leung, 2010). Another small HPC solution provider to enter the cloud market is Penguin Computing with its Penguin on Demand (POD). POD,

according to Penguin Computing, is a high-performance, scalable, on-demand HPC as a Service solution with high-density compute nodes and high-speed storage that offers users a persistent, load on demand, virtualized compute environment (Penguin Computing, 2010).

It is likely that more cloud providers will be moving into this computing area in the future as more people from the scientific community begin to look at this computing service for solutions to their problems.

V. Economics, Flexibility and Green Credentials

Economics, simplification and convenience of the way computing-related services are delivered seem to be among the main drivers of cloud computing (Erdogmus, 2009). Many people see huge potential of the technology in reducing the cost of IT to organizations and freeing them from the expense and hassle of having to install and maintain applications locally (Leavitt, 2009). Providing IT services in the cloud shifts much of this expense to a pay-as-you-go model and consequently offers significant cost advantages according to one view (Lin, Fu, Zhu and Dasmalchi, 2009). Furthermore, a great proportion of the costs of running an IT infrastructure comes from electricity consumption which is needed to run hardware (e.g., PCs, servers, switches, backup drives) and cooling which is needed to reduce the heating generated by the hardware. Cloud computing is likely to reduce expenditure in this area and also reduce labor-related costs, as less people (e.g., technicians) than before will be required to run a cloud-based IT infrastructure.

The cost advantage of cloud computing is not just related to how much cloud users can save by not buying and installing hardware and software and using less power. Users of cloud computing are more likely to significantly reduce their carbon footprint. Research suggests that ICT is already responsible for 2% of global carbon emissions, and that its relative share will increase further. In the UK, for example, increasingly stringent regulations (such as the Carbon Reduction commitment and EU Energy Using Products Directive) are likely to put pressure on educational establishments to make ICT more sustainable (James and Hopkinsons, 2009). In an environment where there is increasing concern about institutions' carbon footprint and energy costs, virtualized services (such as those offered by cloud computing) may become especially appealing (Katz, 2008).

VI. The Growing Appeal of Cloud Computing for HPC

Until recently, high performance computing has not been a good candidate for cloud computing due to a number of factors such as its requirement for tight integration between server nodes via low-latency interconnects and high-speed networking (Shainer et al 2010). For example, the performance overhead associated with host virtualization, a prerequisite technology for migrating local applications to the cloud, quickly erodes application scalability and efficiency in an HPC context which often involves sending messages back and forth many times per second, a process that is likely to increase the possibility of latency (Niccolai, 2009). However, new virtualization solutions that use KVM (kernel-based virtual machine)

and XEN hypervisors[2] have managed to solve the performance issue by reducing the virtualization management overhead through enabling native performance capabilities from the virtual machines (VMs) and by allowing direct access from the VMs to the network. High-speed networking is also an important factor in HPC which requires fast communication between clusters of servers and storage (Shainer et al, 2010). With the ability of more cloud vendors to provider faster speed and networking connections, this issue has become less of a problem for HPC.

Hence, a number of scientific and research communities have begun to look at the possibility of using cloud computing in order to take advantage of its economic and efficiency benefits. The experience of the Medical College of Wisconsin Biotechnology and Bioengineering Center in Milwaukee in the USA is one example. Scientists at this institute are making protein research (a very expensive undertaking) more accessible to scientists worldwide, thanks largely to renting the massive processing power that is available on Amazon's powerful cloud-based servers.

One of the major challenges for many laboratories setting up proteomics programs has been the need to obtain and maintain a computational infrastructure required for analyzing a vast flow of proteomics data generated by mass spectrometry instruments used in determining the elemental composition as well as chemical structure of a molecule. Cloud computing provided that ability at a very competitive cost. This meant that many more users could set up and customize their own systems and investigators could analyze their data in greater depth than was previously attainable, thus making it possible for them to learn more about the systems they are studying (La Susa, 2009; Halligan et al, 2008).

Major cloud computing providers such as IBM and Google are also actively promoting cloud computing as tools for aiding research. In 2007 Google and IBM announced a cloud computing university initiative designed to improve computer science students' knowledge of highly parallel computing practices in order to address the emerging paradigm of large-scale distributed computing. In 2009, the National Science Foundation (NSF) awarded nearly US$ 5 million in grants to fourteen universities through its Cluster Exploratory (CLuE) program to help facilitate their participation in the IBM/Google initiative. The initiative's goal was to provide the computing infrastructure for leading-edge research projects that could help promote better understanding of our planet, bodies and many other issues.

A number of other US government departments are also beginning to explore the merits of cloud computing for scientific purposes. Two of those come to mind: the Department of Energy (DOE) and NASA, the American space agency. The DOE has embarked US$ 32 million for a cloud infrastructure project aimed at exploring the ability of cloud computing to provide a cost-effective and energy-efficient computing service for scientists to accelerate discoveries in a variety of disciplines, including analysis of scientific data sets in biology, climate change and physics. The DOE's centers at the Argonne Leadership Computing Facility (ALCF) in Illinois and the National Energy Research Scientific Computing Center (NERSC) in California are hoping to be able to determine how much of DOE's mid-range computing needs could and should run in a cloud environment and what hardware and software features are needed for science clouds. Due to the exploratory nature of this project, it was named 'Magellan', in honor of the Portuguese explorer who led the first effort to sail

[2] A hypervisor is a virtual machine monitor (VMM) that allows multiple operating system to run concurrently on a host computer.

around the globe and for whom the 'clouds of Magellan' – two small galaxies in the southern sky – were named. Pete Beckman, director of Argonne's Leadership Computing Facility and project lead, said "We know that the model works well for business applications, and we are working to make it equally effective for science." (Argonne National Laboratory, 2009a).

Another interesting cloud-based HPC solution developed by the US DOE at its Argonne National Laboratory[3] is called 'Nimbus' (a Latin word for 'cloud' or 'rain storm'). Nimbus is an open source cloud software infrastructure that allows users to deploy clusters of appliances (i.e., VMs) on remote resources, similar to Amazon's EC2 (see Argonne National Laboratory, 2009b; Keahey, Tsugawa, Matsunaga and Fortes, 2009). The advantages of using cloud computing for scientific purposes were illustrated recently by researchers working on the STAR nuclear physics experiment at Brookhaven National Laboratory's Relativistic Heavy-Ion Collider. The researchers needed simulation results to present at the Quark Matter physics conference but found it difficult to do so as all the computational resources at their disposal were either committed to other tasks or did not support the environment needed for STAR computations. However, the STAR researchers were finally able to use the Nimbus platform in order to dynamically provision virtual clusters on commercial cloud computers and run the additional computations.

Jerome Lauret, software and computing project leader for the STAR project, commented:

> 'The benefits of virtualization were clear to us early on....We can configure the virtual machine image exactly to our needs and have a fully validated experimental software stack ready for use. The image can then be overlaid on top of remote resources using infrastructure such as Nimbus'. (Argonne National Laboratory, 2009b)

He added that with cloud computing 'a 100-node STAR cluster can be online in minutes [whereas] Grid resources available at sites not expressly dedicated to STAR can take months to configure' (Ibid.)

The STAR scientists initially developed and deployed their VMs on a small Nimbus cloud configured at the University of Chicago. Then they used the Nimbus Context Broker[4] to configure the customized cloud into grid clusters which served as a platform for remote job submissions using existing grid tools. However, these resources soon proved insufficient to support STAR production runs.

In order to meet these needs, the Argonne Nimbus team turned to Amazon EC2 for help. The team developed a Nimbus gateway which allowed them to move easily between the small Nimbus cloud and Amazon EC2.

Kate Keahey, the lead member of the Nimbus project, commented:

> 'In the early days, the gateway served as a protocol adapter as well…But eventually we found it easier to simply adapt Nimbus to be protocol-interoperable with EC2 so that the scientists could move their virtual machines between the University of Chicago cloud and Amazon easily' (Ibid.)

[3] The Argonne National Laboratory is operated by the University of Chicago for the United States Department of Energy (DOE).

[4] This Context Broker is a service that allows users to coordinate large virtual cluster launches automatically and repeatedly. It is used to deploy 'one-click' virtual clusters that function right after launch as opposed to launching a set of unconnected' VMs which most VM-on-demand services provide. It also provides a facility to 'personalize' VMs (e.g., enable access policies).

Generating the needed results for the Quark Matter conference at the last minute proved the viability of using cloud resources for scientific purposes, according to Kate Keahey. She elaborated:

> 'One day a provider could be running STAR images, and the next day it could be climate calculations in entirely different images, with little or no effort…With Nimbus, a virtual cluster can be online in minutes." (Ibid.)

The Nimbus cloud is available to all members of the scientific community who want to run their applications in the cloud. To gain access to Nimbus, scientists are required to provide a justification, i.e., a few sentences explaining their science project.

NASA, the US space agency, is also moving towards cloud computing in a big way. It has recently contracted Parabon Computation, a grid computing software company, to deliver a platform that lets NASA scientists and engineers develop and run modeling and simulation applications as a service via a standard Web browser. For example, complex climate models can be constructed that can predict what would happen to global temperatures over the next hundred years if humans double carbon dioxide emissions. According to Michael Seablom, head of the software integration and visualization office at NASA's Goddard Space Flight Center in Maryland:

> 'Trying to get the models running is difficult and it costs us a lot of money here because we have to help groups build the system on their local machine…..The problem with that is if you're a graduate student, you could spend months just trying to get the model running and verify that it's working correctly.' (Brodkin, 2010).

Although the system will initially run climate models, it can be used for many types of scientific research. NASA is also developing its own cloud platform named 'Nebula'. Nebula is a self-service platform built from open source software. It is intended to provide high capacity computing, storage, and network connectivity for NASA's research. Chris Kemp, Chief Information Officer of NASA Ames Research Center, commented:

> 'Nebula has been designed to automatically increase the computing power and storage available to science- and data-oriented web applications as demand rises.' (Miller, 2009).

Interestingly, Nebula's data center is housed in an out-door 40-feet mobile shipping container built by Verari Systems and equipped with Cisco tools. This new approach of building data centers is intended to be a fast and more energy-efficient way of meeting cloud providers' computing and storage requirements (Ibid.).

Nebula is using the open source 'Eucalyptus' software which was developed at the University of California at Santa Barbara. The current interface to Eucalyptus is compatible with Amazon's EC2 interface, thus making it possible for AWS-compatible tools to work with Nebula; a case of promoting inter-cloud operability.

A number of HPC solution providers have begun to take their businesses to the cloud in order to take advantage of the growing demand for cloud-based solutions from the scientific community. Companies such as SGI (with its Cyclone platform) and Penguin Computing

(with its Penguin Computing on Demand), as mentioned earlier, are just two simple examples.

VII. CLOUD PROBLEMS: SHOULD USERS WORRY?

Cloud computing, as indicated above, is an emerging computing service paradigm. Like other new services of this scale and complexity, there are bound to be fears, uncertainties and concerns about the technology's maturity. The most important of those concerns can be identified as those relating to control, vendor lock, performance, latency, security, privacy and reliability.

In some cases, there is an outright rejection of this model. Richard Stallman, creator of the GNU operating system and founder of the Free Software Foundation, described cloud computing as a 'trap' aimed at forcing people to buy into locked, proprietary systems that are likely to prove costly in the future. He once told the Guardian: 'It's stupidity. It's worse than stupidity: it's a marketing hype campaign". This view was also echoed by Larry Ellison, the founder of Oracle, who criticized the 'rash' to cloud computing as 'fashion-driven' and "complete gibberish" and commented that it would be hard to make money in this technology which he sees as 'lacking a clear business model' (Johnson, 2008; Hasson, 2008).

IT managers are likely to be wary of surrendering control of their resources to outside providers who can change the underlying technology without customers' consent. Issues relating to performance and latency (evidenced by the temporary run-outs of capacity by some providers) are also cited as problematic.

Furthermore, there are also valid security and privacy concerns. A recent survey of chief information officers and IT executives by IDC (International Data Corporation) rated security as their main cloud computing concern and almost 75% of respondents said that they were worried about security. Recently, the Electronic Privacy Information Center (a not-for-profit organization), has filed a complaint with the US Federal Trade Commission (FTC) about the security standards of Google's cloud computing, arguing that Google does not encrypt information held on its servers (Marshall, 2009). Moreover, various governments, such as those in the European Union (EU), have privacy regulations that prohibit the transmission of some types of personal data outside the EU. This issue prompted companies such as Amazon and others to develop offerings using storage facilities located in the EU.

Organizations are likely to adopt a careful approach to cloud computing. A survey by EDUCAUSE [5] involving 372 of its member institutions revealed that a great proportion of the respondents with use cases that involved cloud-based services reported that data privacy risks and data security risks were among their top barriers to overcome (Goldstein, 2009).

Another concern is vendor-lock and failures. Currently, many cloud providers offer their services through proprietary APIs. Portability is likely to be increasingly important as the number of cloud providers increases. One solution would be to base those APIs on open source message standards such as SOAP or REST. In some situations this is already happening. For example, Amazon is making its S3 (simple storage service) cloud available through both SOAP and REST and Microsoft ensured that its Azure cloud also supports

[5] EDUCAUSE is a US-based non-profit organization whose mission is to promote the intelligent use of information technology in order to advance higher education.

REST. The need for inter-cloud interoperability was highlighted by Vint Cerf, a co-designer of the Internet's TCP/IP, who likened the current lack of cloud communication standards to that of computer networks in the early 1970s (Krill, 2010). However, there are currently efforts by some organizations such as the Cloud Computing Interoperability Forum to address this issue (Grossman, 2009, p26).

Furthermore, failure of a cloud provider which owns data centers can have serious repercussions for end users who trusted their data with such provider. This issue may force potential cloud users to go for well established and large companies that are likely to be around for many years to come.

Lastly, reliability can also be a serious problem for cloud users. Salesforce.com, for example, left customers without service for six hours in February 2008 while Amazon's S3 and EC2 suffered a three-hour outage in the same month a few days later and an eight-hour outage in July of the same year by S3 (Leavitt, 2009). In early 2009, Google's Gmail (its Webmail service) went down for three hours, thus preventing its 113 million users from accessing their emails or the documents which they store online as "Google Docs" (Naughton, 2009). Vendors often provide service credits for outages. However, those credits, according to a director of a US market research firm, are "cold comfort for sales opportunities missed and executives cut off from business information" (Leavitt, 2009).

Cloud computing may not be suitable for all organizations. For example, for large companies, the loss of service as a result of cloud glitches would be a major concern, particularly if it impacts on their customers and results in substantial loss of sale opportunities and customer dissatisfaction. The issue of reliability with relation to cloud services will continue to be a problem. Similar glitches that befell the cloud services of Amazon and Google are likely to surface again as the number of cloud providers and users increase. However, for small companies struggling to survive the current global economic downturn and cash-strapped educational establishments, often used to similar glitches caused by their old in-house systems, cloud computing is likely to remain an attractive option due to its cost structure and flexibility.

Furthermore, evidence is also emerging to suggest that even large companies, contrary to conventional wisdom, are actually embracing cloud services. A recent report by Forrester (the independent technology and market research company), based upon a survey of small and large enterprises located in North America and Europe, revealed that large firms were more interested than small firms in leveraging IaaS (Infrastructure as a Service) external cloud capability (Golden, 2009).

For organizations involved in scientific research, most of the above concerns may not be as important to them as they might be to those who provide services to consumers e.g., e-commerce companies. For example, the loss of a few hours of services may not be as dramatic for an organization conducting a research experiment as it would be for an online auction or an online chain of retailers. Furthermore, issues of privacy and data protection are likely to be of less concern or indeed relevance, especially if research organizations only use an IaaS cloud for high speed compute operations. So, while the aforementioned cloud drawbacks will remain of concern to many organizations contemplating using cloud computing, their current advantages are likely to outweigh their potential disadvantages for the scientific community. On that basis one could deduce that cloud computing's market share of HPC is more likely to continue rising in the years to come. Indeed, there are speculations that cloud computing's share of the computing service industry as a whole will

increase. In fact, there is an increasingly perceived vision that this computing service will one day be the 5th utility, after water, electricity, gas and telephony (Buyya et al, 2009).

CONCLUSION

Cloud computing is an emerging computing paradigm which promises to provide opportunities for delivering a variety of computing services in a way that has not been experienced before. Until recently, HPC was considered not suitable for cloud computing due to the performance overhead associated with virtualization which can impact on HPC's requirement for tight integration between server nodes via low-latency interconnects. However, these issues have been overcome by virtualization solutions that use new hypervisors such as KVM and XEN. Furthermore, the economic, and indeed the green credentials, of cloud computing have made this computing service an attractive and viable option. To demonstrate the growing popularity of cloud computing for HPC a number of examples were provided in this article of research organizations that decided to go the cloud way.

Despite its growing popularity, there are still some concerns relating to cloud computing such as security, outages and vendor-locking. Given the nascent nature of this technology, this is not surprising. However, there are currently efforts being made to address some of those issues. Furthermore, some organizations are more likely to cope with the consequences of those concerns than others. This article has argued that, given the attractive attributes of cloud computing with relation to 'economy' and 'efficiency', some organizations such as small businesses, academic institutes and those involved in scientific research might find that cloud computing's potential disadvantages are more likely to be outweighed by their current advantages.

REFERENCES

Argonne National Laboratory (2009a), '*DOE to explore scientific cloud computing at Argonne, Lawrence Berkeley national laboratories'*, http://www.anl.gov/Media_Center/News/2009/news091014a.html (Accessed on: 24 January 2010).

Argonne National Laboratory (2009b), '*Nimbus and cloud computing meet STAR production demands*', http://www.hpcwire.com/offthewire/ Nimbus-and-Cloud-Computing-Meet-STAR-Production-Demands-42354742.html?viewAll=y (Accessed on: 5 April, 2010).

Bailey, D (2009), "*Does Azure have the allure to drive enterprises into the cloud*?", *Computing*, http://www.computing 2246131/azure-cure-enterprises-current (Accessed on: 14 March, 2010).

Brodkin, J (2010), *'NASA building cloud service for climate modeling', Computer World,* http://www.computerworld.com/s/article/9150381/ NASA_building_cloud_service_for_climate_modeling?source=rss_software (Accessed on: 4 April, 2010).

Buyya, R., Yeo, C. S. and Venugopal, S. (2009), 'Market-Oriented Cloud Computing: Vision, Hype, and Reality of Delivering IT Services as Computing Utilities', *The 10 th IEEE*

International Conference on High Performance Computing and Communications, pp5-13.

Buyya et al (2009), 'Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility', *Future Generation Computer Systems*, Vol. 25, No. 6, pp. 599-616.

Campbell-Kelly, M (2009), 'The Rise, Fall, and Resurrection of Software as a Service', *Communications of the ACM*, Vol. 52, No. 6.

Campbell, S (2009), 'Timesharing 2.0', *HPCwire*, http://www.hpcwire.com/ specialfeatures/ cloud_computing (Accessed on: 5 April, 2010).

Carr, N (2009), The Big Switch: Re-Wiring the World, from Edison to Google, New York & London: W. W. Norton.

Erdogmus, H (2009), "Cloud Computing: Does Nirvana Hide behind the Nebula?," *IEEE Software*, vol. 26, no. 2, pp. 4-6.

European Commission (2005), The New SME Definition: User Guide and Model Declaration, Enterprise and Industry Publications.

Foley, M. J. (2009), 'Three new codenames and how they fit into Microsoft's cloud vision', *zdnet*, http://blogs.zdnet.com/microsoft/?p=4582 (Accessed on: 15 December 2009).

Golden, B (2009), *'Forrester Bucks Conventional Wisdom on Cloud Computing*', CIO, http://www.cio.com/article/496213/Forrester_Bucks_ Conventional_Wisdom_on_Cloud_Computing (Accessed on: 1 August 2009).

Goldstein, P (2009), 'Alternative IT Sourcing Strategies: From the Campus to the Cloud', *EDUCAUSE Center for Applied Research,* http://net.educause.edu/ ir/library/pdf/ EKF/EKF0905.pdf.

Grossman, R (2009), 'The Case for Cloud Computing', *IT Professional*, Vol. 11, No. 2, pp. 23-27,

Halligan et al (2008), 'Low Cost, Scalable Proteomics Data Analysis Using Amazon's Cloud Computing Services and Open Source Search Algorithms', *Journal of Proteome Research*, Vol. 8, PT. 6, pp. 3148-3153.

Hasson, J (2008), '*Cloud computing is for the birds'*, FierceCIO, http://www.fiercecio.com/story/cloud-computing-birds/2008-10-11 (Accessed on: 5 July, 2009.

James, P and Hopkinsons, L (2009), 'Sustainable ICT in Further and Higher Education: Final Report', *A Report for Joint Information Services Committee* (JISC).

Johnson, B (2008), "Cloud computing is a trap, warns GNU founder Richard Stallman", *The Guardian*, http://www.guardian (Accessed on: 5 July 2009).

Katz, R. N (2008), 'The Gathering Cloud: Is this the End of the Middle?', in Katz, R. N (ed.), The Tower and the Cloud: Higher Education in the Age of Cloud Computing, EDUCAUSE.

Keahey, K., Tsugawa, M., Matsunaga, A and Fortes, J (2009), 'Sky Computing', IEEE Computer Society, September/October.

Krill, P (2010), 'Cerf urges standards for cloud computing', *Info World*, http://www.infoworld.com/d/cloud-computing/cerf-urges-standards-cloud-computing-817 (Accessed on: 5 April, 2010).

La Susa, E (2009), 'Cloud computing brings cost of protein research down to Earth", *Eureka Alert*!, http://www.eurekalert.org/pub_releases/2009-04/mcow-ccb040909.php (Accessed on: 31 July 2009).

Leavitt, N (2009), 'Is Cloud Computing Really Ready for Prime Time?', *Computer*, Vol. 42, No. 1, pp. 15-20.

Leung, L (2010), 'Scientists Offered Free Access to Azure Cloud', *Data Center Knowledge*, http://www.datacenterknowledge.com/archives/ 2010/02/05/scientists-offered-free-access-to-azure-cloud/ (Accessed on: 6 April, 2010).

Lin, G., Fu, D., Zhu, J. and Dasmalchi, G (2009), 'Cloud Computing: IT as a Service', *IT Professional*, vol. 11, no. 2, pp. 10-13.

Marshall, R (2009), "*Privacy group slams Google's cloud services*", V3 (formerly vnunet), http://www.v3.co.uk/articles/print/2238733 (Accessed on: 20 July 2009).

MathWorks (2008), 'Parallel Computing with MATLAB on Amazon Elastic Compute Cloud', *Math Works*, http://www.mathworks.com/products/ techkitpdfs/91593v00.pdf (Accessed on: 6 April, 2010).

Microsoft (2009a), '*Microsoft Case Studies*: RiskMetrics', http://www.microsoft.com/ casestudies/Case_Study_Detail.aspx?CaseStudyID=4000005921 (Accessed on: 8 April, 2010).

Microsoft (2009b), '*SME role for cloud computing'*, http://www. microsoft. com/uk/smallbusiness/sbnews/growing-a-small-business/SME-role-for-cloud-computing-19227631.mspx (Accessed on: 18 July 2009).

Miller, R (2009), 'NASA's Nebula: The Cloud in a Container', *Data Center Knowledge*, http://www.datacenterknowledge.com/archives/2009/12/02/ nasas-nebula-the-cloud-in-a-container/ (Accessed on: 4 April, 2010).

Naughton, J (2009), 'There's silver lining to Google's cloud computing glitch', *The Observer*, http://www.guardian (Accessed on 21 July 2009).

Niccolai, J (2009), 'Penguin puts high-performance computing in the cloud', *Info World*, http://www.infoworld.com/d/cloud-computing/penguin-puts-high-performance-computing-in-cloud-282.

Penguin (2010), '*Penguin Computing on Demand'*, http://www.penguin computing.com/POD/Summary.

Rehr, J. J et al. (2008), '*Scientific Computing in the Cloud'*, http://arxiv.org/ftp/ arxiv/papers/0901/0901.0029.pdf (Accessed on: 10 April, 2010).

Shainer, G et al (2010), '*Cloud Computing Will Usher in a New Era of Science Discovery*', HPC Wire, http://www.hpcwire.com/specialfeatures/ cloud_computing (Accessed on: 10 April, 2010).

Sultan, N (2010), 'Cloud Computing for Education: A New dawn?', *International Journal of Information Management*, Vol. 30, No. 2.

Voas, J and Zhang, J (2009), 'Cloud Computing: New Wine or Just a New Bottle?', *IT Professional*, vol. 11, no. 2, pp. 15-17.

Dr. Nabil Sultan graduated from the University of Liverpool with a PhD in Management in 1992 and in 1996 he received his MSc in Information Systems from the same university. After a two-year working period for the United Nations Development Program (UNDP) as Program Officer he moved on to work as a lecturer in IT and Business for the University of Liverpool's Center for Continuing Education until 1998 and since 1999 as Senior Lecturer and Program Director for Liverpool Hope University's Faculty of Business and Computer Science where he developed many successful modules and programs. Dr. Sultan has many published works in IT and management.

In: Computer Science Research and Technology. Volume 2 ISBN 978-1-61122-073-5
Editor: Janice P. Bauer

Chapter 3

ERROR CONTROL SCHEMES FOR BLUETOOTH TECHNOLOGY: ANALYSIS AND PERFORMANCE

João Henrique Kleinschmidt[1]
Federal University of ABC, Santo André, Brazil

ABSTRACT

This chapter presents different error control schemes of Bluetooth technology, including the standard error control schemes and custom and adaptive error control strategies proposed in the literature. Bluetooth provides an asynchronous connection-less (ACL) link for data transmission. Bluetooth v2.0 with Enhanced Data Rate (EDR) specification offers three transmission rates, which are basic rate of 1 Mbps (types of DHx and DMx), enhanced data rates of 2 Mbps (EDR2, type of 2-DHx) and 3 Mbps (EDR3, type of 3-DHx). Bluetooth employs stop and wait ARQ (Automatic Repeat Request), CRC (Cyclic Redundancy Check) and FEC (Forward Error Correction). The payload contains a header of 1 or 2 bytes and a 2-byte cyclic redundancy code (CRC) for checking the integrity of the field after decoding. In DM (Data-Medium rate) packets the payload is encoded with a (15, 10) shortened Hamming code capable of correcting single bit error in each block. Only one fixed packet type is used during the whole Bluetooth transmission period. These characteristics directly affect the throughput and energy efficiency of a Bluetooth network. The choice of the best packet to be used in a transmission depends of the channel conditions. This chapter describes and analyzes some adaptive schemes proposed in the literature for this purpose. Other issue studied by researchers is to apply different FEC schemes in the Bluetooth packets. Some of recent proposals are described in this paper. An analytical framework is presented to analyze the performance of error control schemes. Performance results are obtained through analysis and simulation in Nakagami-m fading channels, indicating the best scheme for each condition.

[1]joao.kleinschmidt@ufabc.edu.br.

INTRODUCTION

Bluetooth is a low cost wireless radio technology designed to eliminate wires and cables between mobile and fixed devices over short distances, allowing the formation of ad hoc networks [1, 2, 3]. The lower layers of Bluetooth became the IEEE 802.15 standard for wireless personal area networks (WPANs). It operates on the 2.4 GHz ISM (Industrial, Scientific and Medical) band employing a frequency-hopping spread spectrum (FHSS) technique. The transmission rate is up to 3 Mbps. The devices can communicate with each other forming a network (called piconet) with up to eight nodes. Within a piconet, one device is assigned as a master node and the others act as slave nodes. Devices in different piconets can communicate using a structure called scatternet. The channel is divided in time slots of 625 us. A time-division duplex (TDD) scheme is used for full-duplex operation.

In order to improve the reliability of the data sent in the wireless channel, techniques such as automatic repeat request (ARQ) and forward error correction (FEC) can be employed [8]. FEC employs error correcting codes to combat bit errors by adding redundancy (parity bits) to information packets. The receiver uses the parity bits to detect and correct errors. FEC techniques are associated with unnecessary overhead that increases energy consumption when the channel is relatively error free. In ARQ techniques only error detection capability is provided; the receiver requests to the transmitter the retransmission of the packets received in error. Usually an ARQ scheme uses Cyclic Redundancy Check (CRC) codes for error detection [8]. At the receiver, the CRC code verifies the packet. If it detects errors, the node asks a retransmission for the transmitter (negative acknowledgement). If the reception is correct, a positive acknowledgement is sent to the transmitter node. Hybrid ARQ schemes can be developed using the combination of FEC and ARQ schemes. Some typical error control techniques for wireless networks are discussed in [8].

Bluetooth v2.0 [2] with Enhanced Data Rate (EDR) specification offers three transmission rates, which are basic rate of 1 Mbps (types of DHx and DMx), enhanced data rates of 2 Mbps (EDR2, type of 2-DHx) and 3 Mbps (EDR3, type of 3-DHx). The basic rate employs a binary Gaussian Frequency Shift Keying (GFSK) modulation, while EDR2 and EDR3 increase data rates by using Differential encoded Phase Shift Keying (DPSK) modulations, with a constellation of four symbols ($\pi/4$–DQPSK) and eight symbols (8DPSK), respectively. In Bluetooth systems, there are many types of packets. These packets are chosen according to channel conditions [7, 11, 15]. Using a long packet size will increase the throughput of system. A long packet is used in the case of good channels. In the case of bad channels, small packet size is used in transmission process. This will decrease the throughput of system. In the Bluetooth system, there are several channel coding schemes that are implemented. There are three type of error control coding which are used in Bluetooth systems, 1/3 rate error control code, 2/3 rate error control code, and ARQ [1, 2, 3]. Researchers have agreed to the standardization of both access and header fields in Bluetooth. Research concentrates on varying the method of coding the payload which means dividing the payload between data and checksum [4,5,6,7,9,11,12,13,14,15,16,17,18,19].

ERROR CONTROL STRATEGIES FOR BLUETOOTH

Table 1 shows the main characteristics of different ACL packet types provided by Bluetooth v2.0+EDR for asynchronous data transmission [2, 3].

An ACL packet may occupy 1, 3, or 5 consecutive time slots where each slot is assigned 625 u*s*. Bluetooth employs stop and wait ARQ (Automatic Repeat Request), CRC (Cyclic Redundancy Check) and FEC (Forward Error Correction) to achieve high reliability on the hostile wireless links. Bluetooth uses a stop-and-wait ARQ for ACL error control. The receiver will respectively send a positive acknowledgment, ACK (ARQN=1), or negative acknowledgment, NACK (ARQN=0), back to the transmitter for successful or failed packet transmission. Unless the transmitter receives a valid ACK or the retransmission timeout is reached, the packet will be retransmitted during the next transmission [6, 9, 11]. All ACL packet types are composed of three fields, namely access code (AC), packet header and payload. Figure 1 shows the packet format for the basic rate and Figure 2 presents the packet format for the enhanced data rate [1, 2, 3].

Table 1. Asynchronous packet types

Packet	Payload Header (bytes)	User Payload (bytes)	FEC	CRC and ARQ	Symmetric Max. Rate (kbps)	Asymmetric Max. Rate (kbps)	
						Forward	Reverse
DM1	1	0-17	Hamming (15,10)	Yes	108.8	108.8	108.8
DH1	1	0-27	No	Yes	172.8	172.8	172.8
DM3	2	0-121	Hamming (15,10)	Yes	258.1	387.2	54.4
DH3	2	0-183	No	Yes	390.4	585.6	86.4
DM5	2	0-224	Hamming (15,10)	Yes	286.7	477.8	36.3
DH5	2	0-339	No	Yes	433.9	723.2	57.6
AUX1	1	0-29	No	No	185.6	185.6	185.6
2-DH1	2	0-54	No	Yes	345.6	345.6	345.6
2-DH3	2	0-367	No	Yes	782.9	1174.4	172.8
2-DH5	2	0-679	No	Yes	869.1	1448.5	115.2
3-DH1	2	0-83	No	Yes	531.2	531.2	531.2
3-DH3	2	0-552	No	Yes	1177.6	1766.4	235.6
3-DH5	2	0-1021	No	Yes	1306.9	2178.1	177.1

The AC field is used for synchronization, DC offset compensation and piconet identification. The access code is error robust, because the coded synchronization words have a large Hamming distance (d_{min} = 14). The packet header field contains link control information, which includes packet type, destination address, sequence number and acknowledgment flag (ARQN). The header contains a (n,k)=(3,1) repetition code for error verification. A header checksum field (HEC) is used to verify the integrity of the decoded packet header field. The payload field contains a header of 1 or 2 bytes and a 2-byte cyclic redundancy code (CRC) for checking the integrity of the field after decoding. In DM (Data-Medium rate) packets the payload is 2/3 FEC encoded while the DH (Data-High rate) packets do not incorporate any FEC encoding. The FEC scheme employed in DM packets is a (15, 10) shortened Hamming code, which encodes each 10-bit information into a 15-bit codeword

block. This code corrects all single bit errors and detects all two-bit errors in a code word. All ACL packets will be followed immediately by a return packet and therefore the effective number of slots used is 2, 4, or 6. Though the data payloads of DH packets are greater than those of DM packets, the latter packets may achieve higher throughputs in noisy channel since FEC protection can significantly reduce retransmission events. While ACL packets support various transmission rates, only one fixed packet type is used during the whole Bluetooth transmission period. Since it cannot adapt the transmit packets to the dynamic channel conditions, only limited data throughput is expected. To overcome this difficulty, several packet type selection schemes based on channel conditions have been suggested [7, 11, 15, 18].

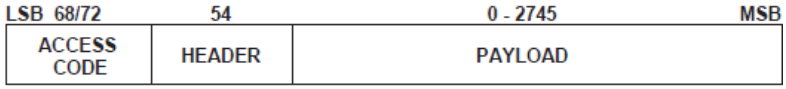


Figure 1. Basic rate packet format.

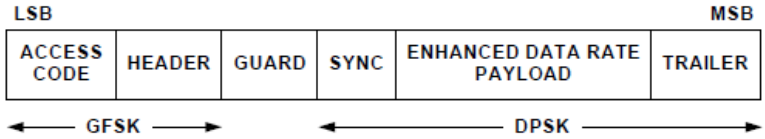


Figure 2. Enhanced data rate packet format.

Whereas the packets defined by the Bluetooth standard (Table 1) have fixed error control schemes, a custom coding can be implemented by making use of the AUX1 packet. With the AUX1 packet the Bluetooth device delivers the received bits independently whether they are correct or not. While the former asynchronous packets with ARQ maintain a reliable link with random delay (which approaches infinity for low values of SNR), the AUX1 packet may alternatively provide an unreliable link with delay of only one time slot.

Valenti and Robert [13] have proposed the use of BCH codes with the CRC code for error detection. As the ARQ is turned off, it must be implemented at the application layer. The coder is implemented by inserting a (232, k) BCH code in the payload of the AUX1 packet. The inputs of the BCH coder are the data and two CRC bytes, resulting in a packet with $K=k-16$ data bits. The code then considered was a (232, 156) binary BCH code with a correction capability of up to $t=10$ errors. In this work five novel modifications in the AUX1 packet are analyzed: (1) BCH code without ARQ; (2) BCH code without ARQ and CRC; (3) Hamming code without ARQ; (4) Hamming code without ARQ and CRC; (5) AUX1 packet with CRC.

The same BCH code of [13] can be applied, but without retransmission (BCH2 and BCH3 packets). Although this strategy can decrease the reliability of transmitted packets, in terms of energy consumption it may be very useful, for it is not necessary to send a return packet to indicate the success of the transmission. The BCH2 packet utilizes the CRC code for error detection, without asking retransmission. A packet is discarded if the CRC detects any errors. The BCH3 packet does not use either retransmission or CRC. The difference between BCH2 and BCH3 is that in the latter the packets are transmitted to the next node (in

a multi-hop network) even if it contains errors, so wasting energy. In the BCH2 packet this fact does not happen, but the packet has additional 16 bits for the CRC implementation.

Another modification is to use the same Hamming code of the DMx packets in the AUX1 payload, but without retransmission, with and without CRC (HAM and HAM2 packets, respectively). Other new packet is the AUX2, which is an AUX1 packet with CRC code. Table 2 shows the error control information for the new introduced packet types.

Table 2. Packet types with custom error control

Packet	Time-slots	Data (bytes)	FEC	ARQ	CRC
AUX2	1	0-27	No	No	Yes
HAM	1	0-18	Hamming (15,10)	No	No
HAM2	1	0-18	Hamming (15,10)	No	Yes
BCH	1	0-17	BCH (232,156)	Yes	Yes
BCH2	1	0-17	BCH (232,156)	No	Yes
BCH3	1	0-17	BCH (232,156)	No	No

ANALYTICAL MODEL

In this section it is presented an analytical model to evaluate the performance of the Bluetooth packets. In order to investigate expressions for performance analysis, a method is used based on [7] and [13] to evaluate the packet error probabilities. A received packet is not accepted when any of the five events happens: (A) the destiny fails to synchronize with the access code of the received packet; (B) the header of the received packet is corrupted (after the repetition code is decoded); (C) the data of the received packet are corrupted after the channel code, if any, is decoded, causing the CRC check to fail; (D) the source is unable to synchronize with the access code of the return packet and (E) the header of the return packet is corrupted.

The synchronization is made correlating the demodulator output with a stored copy of the access code. A packet is synchronized if the correlator output exceeds a given threshold *T*. The frame is synchronized if at least *T* of the 72 bits of the access code were properly demodulated (*T = 65* in this work, the same value used in [7, 13]). The errors are assumed to be independently distributed. The synchronization with the received packet occurs if there are no more than (72 – *T*) errors in the received access code:

$$P[\overline{A}] = \sum_{k=0}^{72-T} \binom{72}{k} \cdot \left[p(\gamma_f)\right]^k \cdot \left[1 - p(\gamma_f)\right]^{72-k},$$

where $p(\gamma_f)$ is the symbol error probability of the forward channel. Since the return packet also has an access code of 72 bits, the probability for the event D has the same form of event A,

$$P[\overline{D}] = \sum_{k=0}^{72-T} \binom{72}{k} \cdot [p(\gamma_r)]^k \cdot [1 - p(\gamma_r)]^{72-k},$$

where $p(\gamma_r)$ is the symbol error probability of the reverse channel. The forward channel is used to send data packets and the reverse channel indicates the success or not of the transmission of a packet (for unidirectional transmission). The events B or E occur if any of the eight triples of the repetition code (3,1) were incorrectly decoded,

$$P[\overline{B}] = \left\{3p(\gamma_f)[1 - p(\gamma_f)]^2 + [1 - p(\gamma_f)]^3\right\}^{18}$$

$$P[\overline{E}] = \left\{3p(\gamma_r)[1 - p(\gamma_r)]^2 + [1 - p(\gamma_r)]^3\right\}^{18}$$

The most probable error is that defined by event C. For DHx, AUX1 and AUX2 packets it occurs when any of the data bytes were received with error:

$$P[\overline{C}] = \left[1 - p(\gamma_f)\right]^b,$$

where *b* is the size of the payload in bits. For DMx and HAMx packets the data are protected by a Hamming code, where B is the number of blocks with 10 bits. The probability of event C for the DMx and HAMx packets is:

$$P[\overline{C}] = \left[15p(\gamma_f)[1 - p(\gamma_f)]^{14} + [1 - p(\gamma_f)]^{15}\right]^B$$

The BCHx packets contain a (232, 156) binary BCH code that can correct up to t=10 errors. Then, for BCHx packets the probability of event C is:

$$P[\overline{C}] = \sum_{k=0}^{t} \binom{232}{k} \cdot \left[p(\gamma_f)\right]^k \cdot \left[1 - p(\gamma_f)\right]^{232-k}$$

Bluetooth basic rate packets uses GFSK modulation with time-bandwidth product *BT=0.5* and modulation index *i* between 0.28 and 0.35. In this work *i=0.32*, the same value used in [7, 13]. When the modulation index *i* is less than 0.5, the signal correlation ρ is given by [10]:

$$\rho = \frac{sin(2\pi i)}{2\pi i}$$

Two constants *a* and *b* can be defined:

$$a = \sqrt{\frac{\gamma}{2}\left(1 - \sqrt{1-\rho^2}\right)},\ b = \sqrt{\frac{\gamma}{2}\left(1 + \sqrt{1-\rho^2}\right)}$$

where γ is the instantaneous signal-to-noise ratio (SNR). The error symbol probability $p(\gamma)$ for the GFSK modulation must be applied and is given by [10]:

$$p(\gamma) = Q_1(a,b) - \frac{1}{2} e^{(a^2+b^2)/2} I_o(ab)$$

where $Q_1\ (a,b)$ is the Q-Marcum function and I_o is the modified Bessel function of first kind. Thus, the packet error probability of the forward channel, PER_f, and reverse, PER_r, can be defined as:

$$PER_f = 1 - \int_0^\infty f(\gamma_f) P[\overline{A}] P[\overline{B}] P[\overline{C}]\, d\gamma_f$$

$$PER_r = 1 - \int_0^\infty f(\gamma_r) P[\overline{D}] P[\overline{E}]\, d\gamma_r$$

where $f(\gamma_f)$ and $f(\gamma_r)$ are the probability density functions and γ_f and γ_r are SNR of the forward and reverse channels, respectively.

The packet retransmission probability then given by

$$P_r(\bar{\gamma}_f, \bar{\gamma}_r) = 1 - \int_0^\infty f(\gamma_f) P[\overline{A}] P[\overline{B}] P[\overline{C}]\, d\gamma_f \cdot \int_0^\infty f(\gamma_r) P[\overline{D}] P[\overline{E}] d\gamma_r$$

The data rate *R* is given by

$$R = \frac{K}{D \cdot N \cdot 625 \cdot 10^{-6}},$$

where *D* is the number of occupied slots per transmission including the reverse packet, *N* is a random variable representing the total number of times a particular packet must be transmitted and *K* is the number of data bits in the packet (DM1:136, DM3: 968, DM5: 1792, DH1: 216, DH3: 1464, DH5:2712, 2-DH1: 432, 2-DH3: 2936, 2-DH5: 5432, 3-DH1: 664, 3-DH3: 4416, 3-DH5: 8168). The average throughput is the expected value of *R* with respect to *N*,

$$\overline{R} = E\{R\} = \sum_{N=1}^{\infty} \left[1 - P_r(\bar{\gamma}_f, \bar{\gamma}_r)\right] \cdot \left[P_r(\bar{\gamma}_f, \bar{\gamma}_r)\right]^{N-1} \cdot R$$

The wireless channel is modeled using the Nakagami fading [10]. This distribution spans, via the fading parameter *m,* the widest range of multipath distributions. When $m \to \infty$, it converges to the AWGN channel and for *m=1* is the Rayleigh fading. Using *m<1* or *m>1* fading intensities more and less severe than Rayleigh are obtained, respectively. The Nakagami probability density function is given by:

$$f(\gamma) = \frac{m^m \gamma^{m-1}}{\Gamma(m)\bar{\gamma}^m} \exp\left(-\frac{\gamma}{\bar{\gamma}}\right), \quad \text{for } \gamma \geq 0$$

where $\bar{\gamma}$ is the average received SNR. It is assumed that the propagation conditions between the transmitter and the receiver are the same in both directions.

The access code and header of 2-DH packet still use the GFSK modulation mode, but the payload of 2-DH packet uses π/4-DQPSK modulation mode. The bit error probability for the access code and header of the forward and reverse packets are computed by the method previously presented. The bit error probability for the payload of the forward packet is computed according to the bit error probability in π/4-DQPSK modulation mode [15]. Compared to GFSK modulation, π/4-DQPSK modulation has higher frequency band utilization ratio. The π/4-DQPSK demodulation of the Bluetooth system can employ the non-coherent demodulation or frequency-discriminating demodulation, thus the hardware is relatively simple to realize, and the system has good ability for decreasing the interference caused by random frequency modulation of fading channel. The theoretical, closed-form representation of bit error probability of the π/4-DQPSK modulation is [10]:

$$p(\gamma) = \left[1 - Q\left(\sqrt{\gamma b}, \sqrt{\gamma a}\right) + Q\left(\sqrt{\gamma b}, \sqrt{\gamma a}\right)\right]\frac{1}{2}$$

where $a = 2 - \sqrt{2}$ and $b = 2 + \sqrt{2}$. This expression is also closely estimated as $p(\gamma) = Q\left(\sqrt{1.1716\gamma}\right)$, where $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-t^2/2} dt$.

Similar to the 2-DH packet, the access code and header of 3-DH packet still use the GFSK modulation mode, but the payload of 3-DH packet uses 8DQPSK modulation mode. The bit error probability for the access code and header of the forward and reverse packets are computed by way of GFSK. The bit error probability for the payload of the forward packet is analyzed according to the bit error probability in 8DQPSK modulation mode. It's difficult to calculate the exact bit error probability expression of the 8DPSK. However the noise variance of DPSK is 200% more than that of 8DPSK [15]. As a result, PSK is 3dB superior to DPSK [10]. According to the definition in [10], let *k*=log2*M*:

$$p(\gamma) \approx \frac{1}{k} 2Q\left(\sqrt{2k(\gamma - 3)} \sin\frac{\pi}{M}\right)$$

where *M*=8, and *k* will be 3. By substituting *M* and *k*,

$$p(\gamma) \approx \frac{2}{3} Q\left(\sqrt{6(\gamma - 3)} \sin \frac{\pi}{8}\right)$$

ADAPTIVE ERROR CONTROL FOR BLUETOOTH MULTI-HOP NETWORKS

Using the same error control scheme for the whole network could be a good choice in some cases, but not always. Sometimes it is needed to apply the best error control available, while in other cases less error control should be used. To use an adaptive error control scheme, a mechanism has to be designed to judge the importance of a packet and then choosing the most efficient error control scheme for that particular packet. In Bluetooth case, to change the error correction scheme means to change the packet type to be transmitted. In order to apply an adaptive scheme in a multi-hop network, where the most important factor is to reduce the energy consumption, an approach presented in [7] was used.

The importance of a packet is evaluated using the multi-hop principle, as shown in Figure 3. The choice of the packet type and the respective error control technique shall be based on the number of hops the packet traveled within the ad hoc network. For instance, a node sends a data packet containing the information of the temperature of a region to the sink node, which collects the data of all the sensors of the network.

However, before the packet reaches the sink node, it may travel through some other nodes of the network with routing capacity. If the packet gets lost at the first hop, only the energy to send the packet from a node to other specific node is lost. If the packet is corrupted after few more hops, much more energy will be spent to transmit the packet through the network. In this sense, a packet is more important if it travels through more nodes in the network, and consequently, more energy is being consumed. An adaptive scheme might use stronger error control techniques for packets that travel more hops and weaker error control for packets with fewer hops.

In the adaptive error control scheme, each packet must have a counter with the number of hops the packet had in the network. This can be implemented as a field in the payload of the packet. Two different adaptive schemes were used: ADP1 and ADP2 [7]. A packet with weaker error control is used for the initial hops and a packet with more powerful coding for the remaining hops throughout the sensor network. Table 3 shows the packet types proposed in these schemes. Although only two schemes are being presented here, other adaptive strategies with different packet types might also be proposed.

Table 3. Adaptive schemes

Scheme	1st and 2nd Hops	3rd, 4th and 5th Hops	Other Hops
ADP1	AUX2	HAM2	DH1
ADP2	AUX2	BCH2	DH1

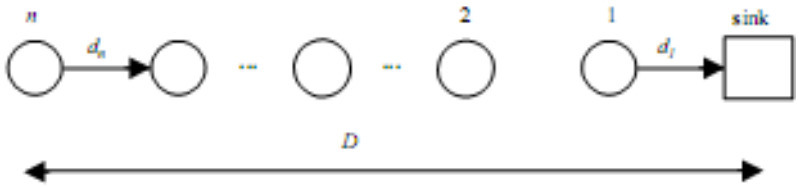


Figure 3. Multi-hop ad hoc network.

The probability of a packet being successfully received at the receiver node is the probability of success of the packet at forward and reverse channels [7]:

$$PA = (1 - PER_f)(1 - PER_r)$$

Thus, the packet error probability for ARQ packets is:

$$PER = 1 - [(1 - PER_f)(1 - PER_r)]$$

This expression can be rewritten as:

$$PER = PER_f + PER_r - (PER_f)(PER_r)$$

The probability of a packet being successfully received at the sink node for the packets without ARQ is:

$$P_{narq} = (1 - PER_f)^H$$

where H is the total number of hops. Let n be the number of retransmissions of ARQ packets. Assuming perfect error detection of the CRC code and infinite retransmissions, the probability that a packet arrives correctly at the sink node is:

$$P_{arq} = \left[\sum_{n_r=1}^{\infty}[p_N[n]]\right]^H = 1$$

The probability of n retransmissions is the product of failure in the n-1 transmissions and the probability of success at the nth transmission:

$$p_N[n] = (1 - PER)(PER)^{n-1}$$

Thus, we evaluate the average number of retransmissions $\overline{N}$ in one hop:

$$\overline{N} = \sum_{n=1}^{\infty} p_N[n] \times n$$

The number of packets that arrive with error at the sink node can be defined for the packets without ARQ as the product of the total number of transmitted packets n_{pac} and the probability that the packet arrives with error at the sink node:

$$n_{error} = (1 - P_{narq}) \times n_{pac}$$

Considering perfect error detection of the CRC code and infinite retransmissions, none of the ARQ packets is received with errors and thus $n_{error} = 0$:

$$n_{error} = (1 - P_{arq}) \times n_{pac} = 0$$

The reliability R is given by the percentage of the sent packets being delivered correctly to the sink node and it may be evaluated as:

$$R = [(n_{pac} - n_{error}) / n_{pac}]$$

Since no specific hardware is being used, the energy consumption in the transmission and reception of the packets is expressed only in normalized terms. While the energies spent in coding and decoding processes were not considered in [11], [12] and [13], in [10] their effect on energy efficiency were shown to be negligible compared to the energy consumed in the transmission of additional parity bits. Thus, only the parity bits of the error control schemes are considered.

The same model of [7] is considered, where the energy consumed per bit is constant and the reception of a determined number of bits consumes approximately 75 per cent of the energy spent to transmit the same number of bits. The minimum energy consumed E_{min} for H hops is evaluated for a packet without error control:

$$E_{\min} = H \times n_{pac} \times \left(n_{aux1} + n_{aux1} \times 0.75\right)$$

where n_{aux1} is the total number of bits of the AUX1 packet. The total energy consumed E in a sensor network for a packet without ARQ and without CRC corresponds to the total number of bits transmitted and received, where each transmitted bit consumes 1 unit of energy and each received bit consumes 0.75 units of energy:

$$E = H \times n_{pac} \times \left(n_{bits} + n_{bits} \times 0.75\right)$$

where n_{bits} is the total number of bits of a packet, including the access code, header and payload.

On the other hand, for the packets with ARQ and CRC and average number of retransmissions $\overline{N}$, the energy E is:

$$E = H \times n_{pac} \times \overline{N} \times \left(n_{bits} + n_{bits} \times 0.75 + n_{ack} + n_{ack} \times 0.75\right)$$

where n_{ack} is the total number of bits of the return packet. In order to evaluate the energy E for the packets without ARQ and with CRC (AUX2, BCH3 and HAM2 packets) the average number of hops has to be computed. The probability that a packet achieves h hops is the product of success in the h-1 hops and the probability of failure in the hth hop, if $h<H$. If $h=H$ the probability of a packet achieving H hops is the product of success in the h-1 hops and the probability of failure in the hth hop added to the probability of success in the H hops:

$$p_H[h] = \left[(1 - PER_f)^{h-1}(PER_f)\right], \text{if h} < \text{H}$$
$$p_H[h] = \left[(1 - PER_f)^{h-1}(PER_f) + (1 - PER_f)^{h}\right], \text{if h} = \text{H}$$

Therefore, the average number of hops $\overline{H}$ can be evaluated as:

$$\overline{H} = \sum_{h=1}^{H} p_H[h] \times h$$

Then the total energy consumed E for the packets with CRC and without ARQ is:

$$E = \overline{H} \times n_{pac} \times \left(n_{bits} + n_{bits} \times 0.75\right)$$

For a Bluetooth ad hoc network to be considered energy efficient, the maximum amount of data bits has to be transmitted with the minimum energy consumption. An energy efficiency parameter η may be defined as:

$$\eta = \frac{E_{\min}}{E} \times R$$

The energy efficiency for an adaptive scheme is evaluated using this equation, but the energy E and the reliability R have to be evaluated in a different manner. For the ADP1 scheme the AUX2 packet is used for the first and second hops, the HAM2 packet for the third, fourth and fifth hops and DH1 packet for the remaining hops of the sensor network. The total energy E is the energy consumed by the different packets:

$$E = E_{aux2} + E_{ham2} + E_{dh1}$$

The energy consumed by the AUX2 packet is:

$$E_{aux2} = \overline{H} \times n_{pac}\left[n_{bits} + n_{bits} \times 0.75\right]$$

where the average number of hops $\overline{H}$ can be evaluated using $H=2$ and n_{bits} is the number of bits of the AUX2 packet. The energy consumed by the HAM2 packet is:

$$E_{ham2} = \overline{H} \times n_{pac}\left[n_{bits} + n_{bits} \times 0.75\right] \times p_{h2}$$

where $\overline{H}$ can be evaluated $H=3$, n_{bits} is the number of bits of the HAM2 packet and p_{h2} is the probability that the AUX2 packet arrives correctly at the receiver after the second hop (because the packet will be discarded if the CRC detects errors):

$$p_{h2} = \left(1 - PER_f\right)^2$$

The total energy consumed by DH1 packet is

$$E_{dh1} = H \times n_{pac} \times \overline{N} \times \left[n_{bits} + n_{bits} \times 0.75\right] \times p_{h5}$$

where H is the number of the remaining hops of the network, n_{bits} is the number of bits of the DH1 packet and p_{h5} is the probability that the HAM2 packet arrives correctly at the receiver after the fifth hop, given by:

$$p_{h5} = \left(1 - PER_f\right)^3 \times p_{h2}$$

The number of transmitted packets with error is the sum of errors occurred in the transmissions of the AUX2 and HAM2 packets, as the DH1 packet will always be retransmitted until it is correctly received:

$$n_{error} = n_{error\_aux2} + n_{error\_ham2}$$

The number of errors of the AUX2 packet is the product of the total number of transmitted packets n_{pac} and the probability of error of the AUX2 packet in two hops:

$$n_{error\_aux2} = [1 - (1 - PER_f)^2] \times n_{pac}$$

The number of errors of the HAM2 packet has the same form, but the number of errors occurred in the first and second hops using the AUX2 packet have to be subtracted from the total number of transmitted packets n_{pac}:

$$n_{error\_ham2} = [1 - (1 - PER_f)^3] \times (n_{pac} - n_{error\_aux2})$$

Then the reliability and the energy efficiency can be evaluated. For the ADP2 scheme, the energy efficiency may be evaluated using the same equations of ADP1 scheme, replacing the HAM2 packet by the BCH2 packet.

RESULTS

Figures 4 and 5 show the results obtained for the different packet types with fading parameter $m=\infty$ (AWGN channel). For each channel condition, a different packet has the best performance. Bluetooth specification does not suggest any scheme indicating how to select packet type during communication. Each link can employ different packet type to transmit data information. Due to the different packet length, error-check and error correcting schemes, each packet has the different retransmission probability when it has the different average SNR [15]. From the results, the following adaptive packet selection scheme can be observed: when SNR > 15.2 dB, the 3-DH5 packet achieves the highest throughput and can be selected. For 10.5dB < SNR < 15.2dB, select the 2-DH5 packet, for 9.7dB < SNR < 10.5 the 2-DH3 is the best packet and finally, for SNR < 9.7 dB, select the 2-DH1 packet [15].

In the evaluation of energy efficiency a sensor sends 100.000 packets to the sink (n_{pac}=100.000), considering different number of hops [7]. The value of the Nakagami fading parameter is m=1. The data is some variable that could be transmitted with few bytes of data. The data size to be transmitted was chosen to be 17 bytes. Although other data sizes could be used, this value may indicate a tendency of the packet performance. The value of 17 bytes was chosen because it is the maximum number of data bytes that the DM1 and BCH packets can transmit. Figures 6 to 8 show the results obtained for the energy efficiency of some packets as a function of signal-to-noise ratio, for different number of hops. Only the main packets are shown in the graphs. For a single hop network (Figure 6) the AUX1 packet has the best efficiency for SNR values higher than 15 dB, approximately. When the SNR is below this value, the BCH3 packet is the best.

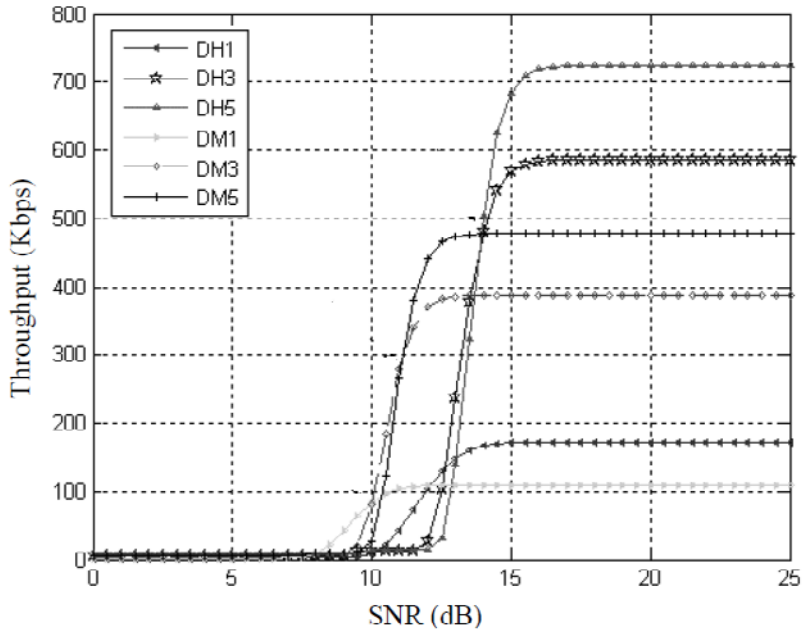


Figure 4. Throughput for basic rate data packets.

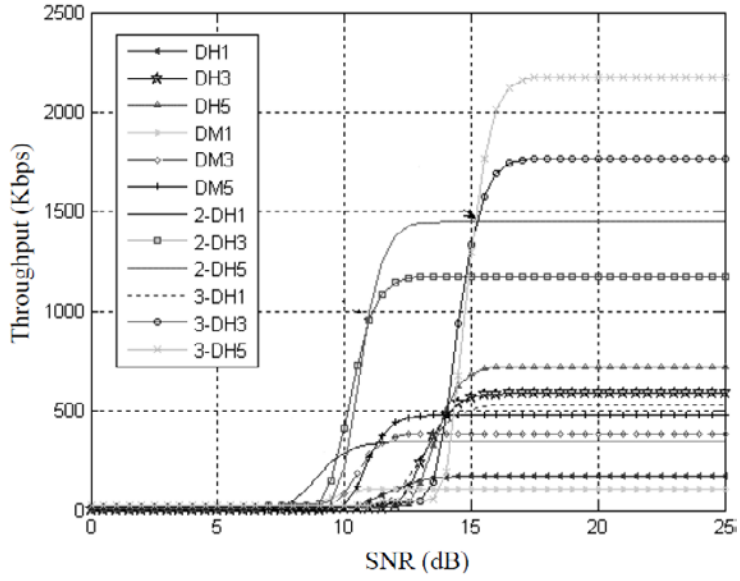


Figure 5. Throughput for all data packets.

With 10 hops (Figure 7) the relative performance among the packets begins to stabilize. The AUX1 packet only has the higher efficiency for channel conditions above 30 dB. For approximately 30 dB the AUX2 packet becomes the best. The adaptive scheme ADP2 has the best efficiency when the SNR is close to 20 dB and the BCH packet is the best for SNR below 15 dB. It can be noted that when the channel quality is good, it is not necessary a very powerful error correction and the AUX1 and AUX2 packets can be utilized. If the channel conditions are very bad, a code able to correct many errors has to be used, so the BCH packet is the most recommended in such situations. For intermediary conditions, the adaptive schemes ADP1 and ADP2 have the best energy efficiency degree. This behavior of the different error control strategies is approximately the same for 25 hops (Figure 8) [7].

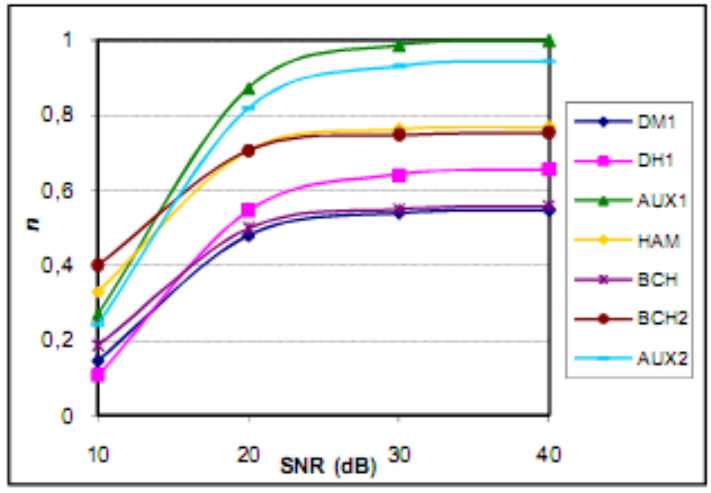


Figure 6. Energy efficiency for 1 hop.

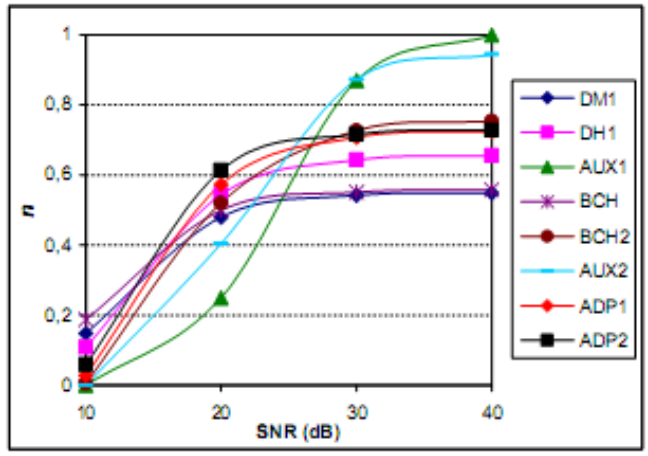


Figure 7. Energy efficiency for 10 hops.

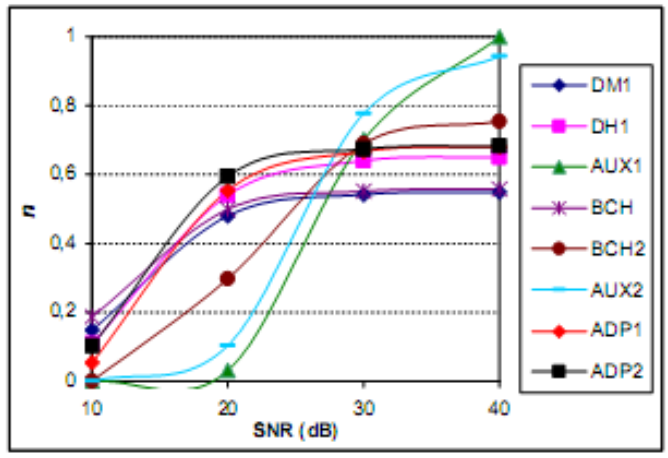


Figure 8. Energy efficiency for 25 hops.

CONCLUSION

In this work the performance of asynchronous data packets defined by Bluetooth specification was studied. Each packet use different error control schemes for error correction and detection. We analyze the performance of the standard Bluetooth packets, custom coding and adaptive error control schemes.

An analytical model was presented to evaluate the throughput and energy efficiency of Bluetooth data packets in Nakagami-*m* fading channels. The results have shown that for good channel conditions the packets with little or no error protection present the best performance. For bad channel conditions, packets with some stronger error correction scheme have the best performance. In the case of a network designer who has information about the channel conditions, these considerations may help in what scheme to use for each situation.

REFERENCES

[1] Bluetooth Special Interest Group. (2001). *Specification of Bluetooth System,* ver. 1.1.

[2] Bluetooth Special Interest Group. (2004). Specification Volume 2: Core System Package, *Bluetooth specification Version 2.0 + EDR.*

[3] Bluetooth Special Interest Group. (2009). Covered Core Package, *Bluetooth specification Version 3.0 + HS.*

[4] Chen, L. J.; Sun, T.; Chen, Y. C. (2006). Improving Bluetooth EDR data throughput using FEC and interleaving. *Proc. of the Second International Conference on Mobile Ad Hoc and Sensor Networks*, p. 1-12.

[5] Fei, X.; Yiqi, Z. A method to enhance the throughput performance of Bluetooth 2.0+EDR specification. (2007*). Proc. of IEEE International Conference on Wireless Communications, Networking and Mobile Computing*, p. 2132-213.

[6] Hipólito, J. I.; Arballo, N. C.; Macarty, A. M.; Garcia, E. J. (2009). Bluetooth performance analysis in wireless personal area networks. *Electronics, Robotics and Automotive Mechanics Conference*, p. 38-43.

[7] Kleinschmidt, J. H.; Borelli, W. C.; Pellenz, M. E. (2009). An energy efficiency model for adaptive and custom error control schemes in Bluetooth sensor networks. AEU – *International Journal of Electronics and Communications*, vol. 63, p. 188-199.

[8] Liu, H.; Ma, H.; El Zarki, M.; Gupta, S. (1997). Error control schemes for networks: an overview. *Mobile Networks and Applications*, vol. 2, p. 167-182.

[9] Perrucci, G. P.; Pedersen, M. V.; Madsen, T. K.; Fitzek, F. H. P. (2009). Energy evaluation for Bluetooth link layer packet selection scheme. *European Wireless*, p. 250-254.

[10] Proakis, J. Digital Communications. New York, NY, McGrawHill, 3rd edition, 2001.

[11] Shih, C. H.; Wang, K.; Shih H. C. (2006). An adaptive Bluetooth packet selection and scheduling scheme in interference environments, *Computer Communications*, vol. 29, p. 2084–2095.

[12] Tan, T. H;, Huang Y.; Chiu H., Chang, C. (2009) . Enhancing Bluetooth data throughput using a mode/packet switching scheme based on arqn statistics. Proc. of the Eighth International Conference on Machine Learning and Cybernetics, *Baoding*, p. 2916-2921.

[13] Valenti, M. C.; Robert, M. (2002). Custom coding, adaptive rate control and distributed detection for Bluetooth. *IEEE Vehicular Technology Conference,* p. 918-22.

[14] Valenti, M. C.; Robert, M. (2002). Improving the QoS of Bluetooth through turbo coding. *IEEE Military Communications Conference.*

[15] Yang, F; Ke, W.; Zhihong, Q. (2005). Performance analysis of Bluetooth packet transmission and adaptive packet selection strategy. *Journal of China Institute on Communications.*

[16] Yufeng, N.; Yiqi, Z.; Naikuo, C. (2009). Analysis of packet interference and aggregated throughput of bluetooth piconets. *International Conference on Networks Security, Wireless Communications and Trusted Computing*, p. 129-132.

[17] Zanella, A.; Zorzi, M. (2008). Throughput and energy efficiency of Bluetooth v2 + EDR in fading channels. *Proc. of IEEE Wireless Communications and Networking Conference*, p. 1661-1666.

[18] Zanella, A. (2009). Carrier–sense ARQ: squeezing out Bluetooth performance while preserving standard compliancy. *Proc. IEEE Conference on Communications.*

[19] Zanella, A. (2009). A mathematical framework for the performance analysis of Bluetooth with enhanced data rate. *IEEE Transactions on Communications,* vol. 57, no. 8, p. 2463-2473.

In: Computer Science Research and Technology. Volume 2 ISBN 978-1-61122-073-5
Editor: Janice P. Bauer

Chapter 4

POSSIBILITY OF HIGH PERFORMANCE COMPUTATION IN BIOLOGICAL BRAINS

Takaaki Musha
Technical Research and Development Institute, MOD, Japan

ABSTRACT

The quantum processor is considered to have the possibility to overcome fundamental limitations of conventional processors, but the influence of energy cost due to the uncertainty principle may prevent speeding up the quantum computation. On the basis of the theorem that the evanescent photon is a superluminal particle, the possibility of a high performance computer system compared with conventional silicon processors has been studied and it can be shown that the energy loss required for computation is much lower than the conventional processor by utilizing evanescent photons. From the high performance capabilities of quantum computation, there are many researchers for explaining the higher performance of human brains, including consciousness. R.Penrose and S.Hameroff have proposed the idea that the human brain can attain high efficient quantum computation by functioning of microtubular structure of neurons. But Tegmark estimated the duration of coherence to be on the order of 10^{-13} seconds, which is far smaller than the one tenth of a second associated with consciousness and it is normally expected that any macroscopic coherence of a quantum state in a warm wet brain to be destroyed almost immediately. Hagen, Hameroff, and Tuszynski have claimed that Tegmark's assumptions should be amended, but the need to maintain macroscopic quantum coherence in a warm wet brain is certainly a serious problem for the Penrose-Hameroff model. From the assumption that the evanescent photon is a superluminal particle, it can be also shown that the microtubule in biological brain can achieve large quantum bits computation compared with the conventional silicon processors, which leads to the high performance computation compared with the conventional processors even at the room temperature, contrary to the Tegmark's calculation result. If this mechanism is true for the brain function, we can obtain the possibility to realize much more efficient computer systems like a human brain by utilizing superluminal evanescent photons for quantum processors.

INTRODUCTION

Moore's law describes the trend of computing hardware, in which the performance of the microprocessor has doubled every two years as shown in Figure 1. But It is widely believed that Moore's law for microprocessor performance will fail to hold in the next decade due to the brick wall arising from fundamental physical limitations of the computational process.

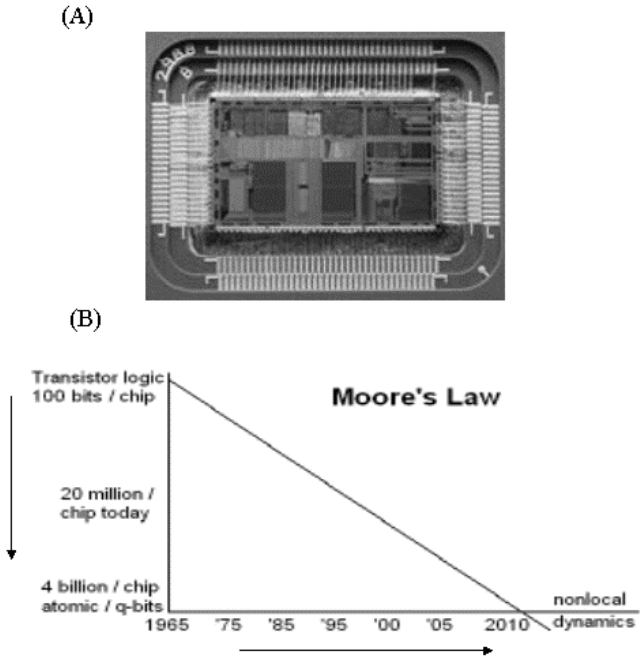


Figure 1. Logic gate in the PC (A) and the Moore's law (B).

Instead of conventional computer systems, a quantum computer would store information as either 1 and 0, or a quantum superposition of the two states. Such a "quantum bit," called a qubit, allows for far greater flexibility than the binary system of conventional computers. Specifically, it is considered that a quantum computer would be able to perform calculations on a far greater order of magnitude than traditional computers.

However, R.Feynman[1986] discussed the possibility of a quantum computer by taking an example of reversible computing in his paper and he pointed that computational energy cost versus speed was limited by the energy dissipation during computation. According to Feynman's idea, the computational speed is limited by minimum energy required to transport a bit of information irreversibly between two devices, which prevents the speeding up of quantum computation.

From the assumption that the evanescent photon is a superluminal particle called a tachyon, the author studies the possibility of realization of a high performance computing system by utilizing superluminal evanescent photons and he has also studied the possibility that the microtubular structure of neurons in a human brain is functioning as a quantum computational system that can attain higher efficient computation compared with conventional silicon processors.

POSSIBILITY OF SUPERLUMINAL SPEED IN QUANTUM REGION

E.Recami [2001] claimed in his paper that tunneling photons traveling in an evanescent mode can move with superluminal group speed, which can be shown as follows;

The evanescent photon generated in quantum domain satisfies the following Klein-Fock-Gordon equation given by

$$\left[-\frac{1}{c^2}\frac{\partial^2}{\partial t^2}+\nabla^2-\frac{m_*^2c^2}{\hbar^2}\right]A(x,t)=0 \tag{1}$$

where c is the light speed, m_* is an absolute value of the proper mass of the evanescent photon and $\hbar$ is the Plank constant divided by 2π.

This equation has the solution for the photon traveling in an evanescent mode given by

$$A(x,t)=A_0\exp\left[-\frac{Et+px}{\hbar}\right] \tag{2}$$

which corresponds to the elementary particle with an imaginary mass im_* that travels at a superluminal speed satisfying

$$E^2=p^2c^2-m_*^2c^4 \tag{3}$$

where E is the energy of the superluminal particle and p is its momentum given respectively as

$$E=\frac{m_*c^2}{\sqrt{v^2/c^2-1}} \tag{4}$$

and

$$p=\frac{m_*v}{\sqrt{v^2/c^2-1}} \tag{5}$$

Hence, it can be seen that tunneling photons traveling in an evanescent mode can move at a superluminal speed.

By the pulse propagation experiment, Chu and Wong at AT&T Bell Labs measured superluminal velocities for light traveling through the absorbing material [Brown,1995]. Furthermore Steinberg, Kwait and Chiao [1993] measured the tunneling time for visible light through the optical filter consisting of the multilayer coating about 10^{-6} m thick.

Measurement results by Steinberg and co-workers have shown that the photons seemed to have traveled at 1.7 times the speed of light. Recent optical experiments at Princeton NEC have also verified that superluminal pulse propagation can occur in transparent media [Wang et al.,2000]. These results indicate that the process of tunneling in quantum physics is superluminal as claimed by E.Recami, which may be applied for realization of the high performance computer system.

SIGNAL PROCESSING BY THE TUNNELING PHOTON

Recently it has been discovered by the research team at the University of Maryland that a signal processing devise based on single photon tunneling can be realized [Smolyaninov et al.,2004]. They conducted the experiment for the transmission of light through nanometer-scale pinholes in a gold film covered by a nonlinear dielectric, which revealed that transmittance of a nanopore or nanopore array at one wavelength could be controlled by illumination with a second, different, wavelength. This opens the door to optical signal processing devises, such as all-optical switches realized on a microscopic scale which can manipulate single electrons, atoms or photons. If the atoms can be localized to distances small compared to the radiation wavelength, they can be coherently coupled by photons and an entangling quantum logic gate can be realized.

By applying this technology, the computer gate consisting of a large number of small, interconnected electromagnetic ion traps created for both memory and logical processing by manipulating atoms with the tunneling photon may be realized as shown in Figure 2.

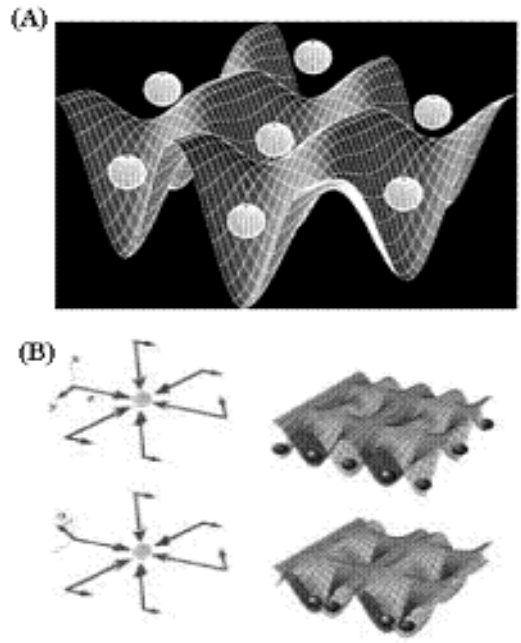


Figure 2. Optical lattice (A) and Quantum logic performed in a three-dimentional optical lattice (B).

The possibility of nano-switching by using evanescent photons in optical near-field was also proposed by T. Kawazoe and his co-workers [Whipple, 2002].

By utilizing this technology, it is considered that the computation utilizing quantum tunneling photons has the possibility to achieve much faster computational speed and the influence of energy cost due to the uncertainty principle which prevents speeding up the quantum computation can be overcome shown as follows.

UNCERTAINTY PRINCIPLE FOR SUPERLUMINAL ELEMENTARY PARTICLES

The uncertainty principle for superluminal particles can be obtained shown as follows;

From relativistic equations of energy and momentum of the moving particle, given by

$$E = \frac{m_0 c^2}{\sqrt{1 - v^2 / c^2}} \tag{6}$$

and

$$p = \frac{m_0 v}{\sqrt{1 - v^2 / c^2}} \tag{7}$$

we can obtain the relation $p / v = E / c^2$.

From which, we have

$$\frac{v\Delta p - p\Delta v}{v^2} = \frac{\Delta E}{c^2} \tag{8}$$

Supposing that $\Delta v / v^2 \approx 0$, Eq.(8) can be simplified as

$$\Delta p \approx \frac{v}{c^2} \Delta E \tag{9}$$

This relation can be also applied for the superluminal particle, which energy and the momentum can be shown in Eqs. (4) and (5) respectively [Musha, 2006].

According to M.Park and Y.Park [1996], the uncertainty relation for the superluminal particle can be given by

$$\Delta p \cdot \Delta t \approx \frac{\hbar}{v - v'} \tag{10}$$

where and v and v' are the velocities of a superluminal particle before and after the measurement, and Δt is a time interval.

When we let $v' = c$ and $\beta = v/c$, Eq.(10) can be rewritten as

$$\Delta E \cdot \Delta t \approx \frac{\hbar}{\beta(\beta - 1)} \tag{11}$$

from Eq.(9), which gives the uncertainty relation between energy and time for superluminal particles.

ENERGY COST FOR QUANTUM COMPUTATION UTILIZING TUNNELING PHOTONS

Hereafter we assume that tunneling photons which travel in an evanescent mode can move with a superluminal group speed as confirmed by some experimenters.

We consider the computer system which consists of quantum gates utilizing quantum tunneling photons to perform logical operations. Benioff showed that the computation speed was close to the limit by the time-energy uncertainty principle [Benioff,1982].

Margolus and Levitin extended this result to a system with an averaged energy $< E >$, which takes time at least $< E >= \pi\hbar/(2\Delta t)$ to perform logical operations [Margolus and Levitin, 1998; Lloyd, 2000].

Summing over all logical gates of operations, the total number of logic operations per second is no more than

$$\Delta t \approx N \frac{\pi \hbar}{2 < E >} \tag{12}$$

where Δt is a operational time of an elementary logical operation and N is the number of consisting gates of the computer. From Eq.(12), energy spread for the quantum tunneling photon (abbreviated QTP hereafter) gate becomes $1/\beta(\beta - 1)$ times the energy spread for the logical gate using particles moving at sub-luminal speed including photons, then the total number of logic operations per second for QTP gates can be given by

$$\Delta t \approx \frac{N}{\beta(\beta - 1)} \frac{\pi \hbar}{2 < E_* >} \tag{13}$$

where $< E_* >$ is an averaged energy for QTP gates.

As an uncertainty in the momentum of tunneling photons moving at the superluminal speed can be given by

$$\Delta p = \frac{m_* v}{\sqrt{v^2 / c^2 - 1}} - \frac{\hbar\omega}{c} \tag{14}$$

where m_* is an absolute value of the mass for the tunneling photon moving at superluminal speed and ω is an angular frequency of the photon, the velocity of the tunneling photon can be estimated as

$$v \approx c\left(1 + \frac{1}{\sqrt{\omega \Delta t}}\right) \tag{15}$$

from Eqs.(4) and (10), and $E = \hbar\omega$ [Musha, 2006].

If we let the tunneling distance be d, the time for a photon tunneling across the barrier can be roughly estimated as by $\Delta t = d / v$. Then the velocity of the tunneling photon can be given by

$$v \approx c\left(1 + \frac{c}{2\omega d} + \sqrt{\frac{c}{\omega d} + \frac{c^2}{4\omega^2 d^2}}\right) \tag{16}$$

From which, the ratio of the minimum energy required for computation by QTP gates and the conventional computation can be given as follows by equating Eqs.(12) and (13);

$$R = \frac{< E_* >}{< E >} \approx \frac{1}{\beta(\beta - 1)} \tag{17}$$

where

$$\beta \approx 1 + \frac{c}{2\omega d} + \sqrt{\frac{c}{\omega d} + \frac{c^2}{4\omega^2 d^2}} \tag{18}$$

By the Higgs mechanism in quantum field theory, the penetration depth of tunneling photons can be estimated by [Jibu et al., 1996].

$$r_0 = \frac{\hbar}{Mc} \tag{19}$$

where M is the effective mass which yields $M \approx 10eV$.

From which, the penetration depth of the tunneling photon is estimated as 1.5×10^{-8} m. Then the ratio of minimum energy required for the computation by QTP gates to the conventional computation processes can be estimated as shown in Figure 3, when we let the tunneling distance of the barrier be $d \approx 10n$ m.

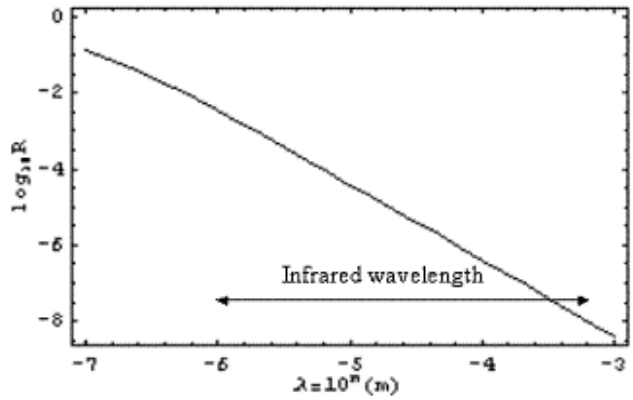


Figure 3. Ratio of the minimum energy require for the computation.

In this figure, the horizontal axis is for the wavelength of tunneling photons and the vertical axis is for the ratio of their energy required for the computation.

If the wavelength of the tunneling photon is in a far infrared region $(\lambda = 10^5 nm)$, the energy cost of computation for the computer, which consists of QTP gates, reaches to 10^{-6} times smaller than that of conventional computer systems.

In recent years, many studies on the quantum computation were conducted and it was recognized that the computational speed by quantum computing was much higher than that of conventional silicon processors. But the energy cost due to the uncertainty principle which prevents speeding-up of the quantum computation was not considered.

From the theoretical analysis of the energy limit of the quantum computer system which utilizes tunneling photons, it can be shown that energy loss of computation by utilizing superluminal tunneling photons is much lower than that of conventional silicon processors. Moreover this superluminal effect can eventually speed up computers significantly because it can compensate interconnect delays inside logic gates which can never be fully eliminated from any real electronic components and bringing overall transmission rate closer to the ultimate speed limit [Chiao et al., 2001], which actually boost the speed of a signal traveling on an electromagnetic wave for achieving high performance computers.

DIFFICULTIES WITH QUANTUM COMPUTATION

Another problem for conducting quantum computation is decoherence of quantum states. The qubit calculations of the quantum computer are performed while the quantum wave function is in a state of superposition between states, which is what allows it to perform the calculations using both 0 and 1 states simultaneously.

But this coherent state is very fragile and difficult to maintain. By the slightest interaction with the external world would cause the system to decohere. This is the problem of decoherence which is a stumbling block for quantum computation. Therefore the computer has to maintain the coherent state for making calculations.

If we let T is the relaxation time of a single qubit and Δt is the operation time of a single logical gate, the figure of merit γ for computation can be defined as $\gamma = T/\Delta t$ [Haroche and Raimond, 1996], which is on the order of the number of qubits times the number of gate operations. As a superposition state of the L-qubits system would cause decoherence approximately 2^L times faster than a superposition state of one qubit [Gea-Banacloche, 2005], then the relaxation time of the L-qubits system can be roughly estimated to be 2^{-L} times the relaxation time of a single qubit computation. Thus the minimum energy required to perform quantum computation for the L-qubits system can be given from Eq.(11) as

$$E_0 \approx \frac{\hbar \nu_G L}{T} 2^L \tag{20}$$

where ν_G is the number of gate operations.

Similar to this equation, the minimum energy required to perform quantum computation utilizing superluminal particle can be estimated as [Musha,2009]

$$E_0' \approx \frac{\hbar \nu_G L}{\beta(\beta-1)T} 2^L \tag{21}$$

Supposing that $E_0 = E_0'$, an increase of qubit size to perform computation by superluminal evanescent photon compared with the conventional computation can be given by

$$\Delta L \approx \frac{\log_2[\beta(\beta-1)]}{1+1/L\log 2} \tag{22}$$

when satisfying $\Delta L < L$.

From which, we can estimate an increase of qubit size of the quantum computation utilizing superluminal evanescent photons compared with the conventional computer system.

POSSIBILITY OF QUANTUM COMPUTATION IN THE BIOLOGICAL BRAIN

From the high performance capabilities of quantum computation, there are many researchers for explaining the higher performance of human brains, including consciousness. As Feynman proposed that the optical computing network is the most realizable quantum mechanical computer among many possibilities such as a superconducting computer and Hameroff suggested that microtubles in the brain were acting as waveguides for photons and as holographic processors [Jibu et al, 1994].

The cytoskelton of biological cells, including neurons of the brain, is made up of microtubles as shown in Figure 4.

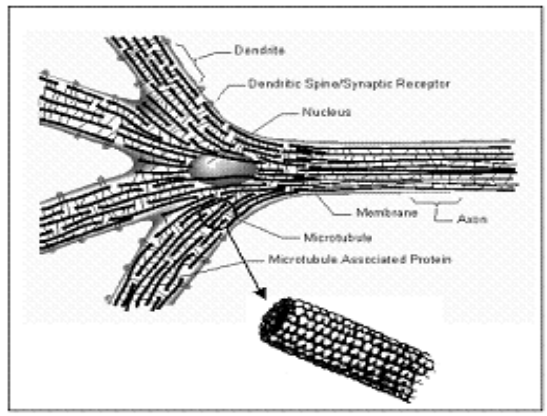


Figure 4. Cytoskelton of biological cells made up of microtubules.

Each microtuble is a hollow cylindrical tube of tublin proteins as shown in Figure 5, which outer core diameter is 25 nm.

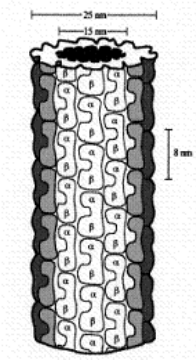


Figure 5. Hollow cylindrical polymers of tublin proteins.

Microtubules are comprised of subunits of the protein, named tubulin [Satinover,2001]. Proteins contain hydrophobic (water repellent) pockets and these pockets contain atoms with electrons called π electrons, which means electrons in the reactive outer part (outer shell) of the atom that are not bonded to other atoms. The tubulin protein subunits of the microtubules have hydrophobic pockets within two nanometers of one another, which is close enough for the π electrons of the tubulin to become quantum entangled.

Each tublin molecule can function as a switch between two conformations which exist in quantum superposition of both conformational states.

As the dimensions of centrioles are close to the wavelengths of light in the infrared and visible spectrum such that they may act as phase coherent, resonant waveguides [Albrecht-Bueler,1992] as shown in Figure 6, which can be mapped onto the spin of material and they may play the role of retrieval, coupling/entangling of spin states of materials [Hameroff,2004].

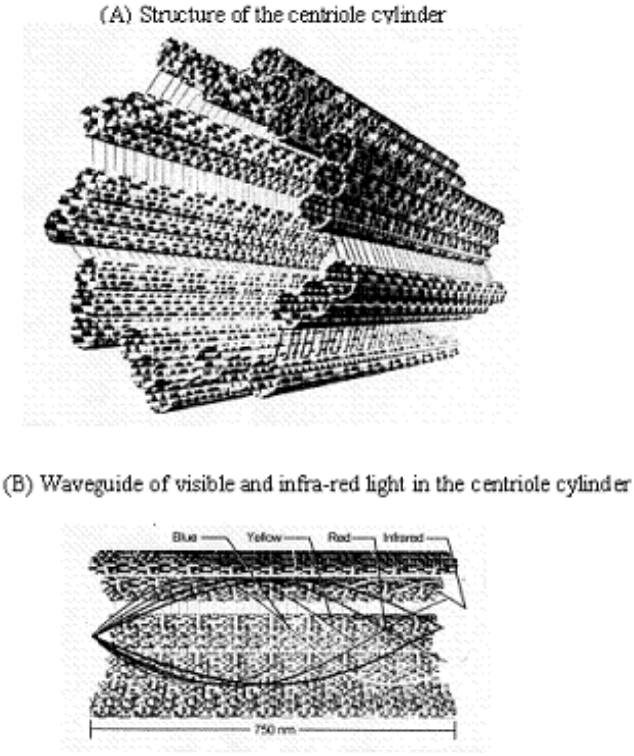


Figure 6. Structure of the centriole cylinder (A) and the wavelength of light suggesting possible waveguide behavior (B) (Hameroff, 2004).

Thus each tublin molecule can function as a switch between two conformations which exist in quantum superpositions of both conformational states.

According to Jibu et al. [1996,1997a,b], microtuble quantum states link to those of other neurons by quantum coherent photons tunneling through membranes in biological systems and the cytoskeletal protein conformational states are entangled by these photons which form coherent domains in their interaction with the local electromagnetic field.

Hameroff and Tuszynski [2004] have proposed that microtuble subunit tublins undergo coherent excitations, which leads to the automatic sequence where quantum coherence superposition is emerged in certain tublins and consciousness is occurred as the process shown in Figure 7. According to their hypothesis of quantum brain, microtuble quantum states link to those of other neurons by quantum coherent photons tunneling through membranes in biological systems functioning in a way resembled as an ion trap computers.

On the other hand, Hameroff and Penrose [1996] have constructed a theory, in which human consciousness is the result of quantum gravity effects in microtubules, which they dubbed Orch-OR (orchestrated object reduction) that involves a specific form of quantum computation conducted at the level of synapses among brain neurons. They have suggested that microtubles in brain neurons function as quantum computers with tublin proteins in macrotubles acting as a quantum bits of computation through two-state q-bits formed from tublin monomers.

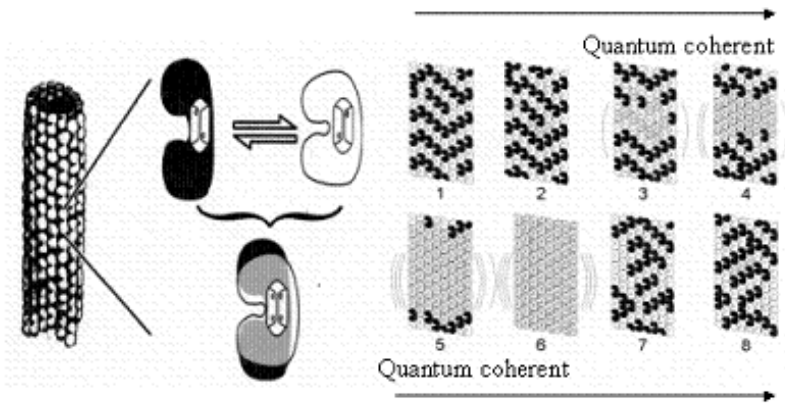


Figure 7. Quantum coherence superposition in tublins.

M.Jibu et al. [1994, 1996, 1997a,b] also proposed that the conscious process in the brain was related with the macroscopic condensates of massive evanescent photons generated by the Higgs mechanism, i.e. that of general biological cell functioning arising from dynamical effects of electromagnetic interaction among electric dipoles in biological systems, which were studied by Del Guidance et.al [1986]. They claimed that human consciousness could be understood as arising from those creation-annihilation dynamics of a finite number of evanescent (tunneling) photons in the brain. They also considered that each microtuble was a coherent optical encoder in a dense microscopic optical computing network in the cytoplasm of each brain cell, acting as a holographic information processor. G.Vitiello [1995] proposed that consciousness mechanisms in the brain was achieved by the condensation of collective modes (called Nambu-Goldstone bosons) in the vacuum.

Faber et al [2006] proposed the models of the mind based on the idea that neuron microtubles could perform computation and they estimated the favorable condition for storage and information processing, which was found at temperatures close to the human body.

However, it was found by Unruh [1995] that the time required in quantum computation must be less than the thermal time scale $\hbar / k_B T$, which yields 2.6×10^{-14} sec at the room temperature ($=20°C$). This value is too small for the conventional silicon processors to conduct a single gate operation.

Max Tegmark [2000] also published a refutation of the Orch-OR model in his paper that the time scale of neuron firing and excitations in microtubles was slower than the decoherence time by at least $1/10^{10}$. According to his paper, it is reasonably unlikely that the brain functions as a quantum computer at room temperature.

But some researches suggest that the brain requires quantum computing for perception [Bialek and Schweitzer, 1987], which means that the human brain has to work as a quantum processing system. Hameroff [1982] proposed the idea in his paper that the biological information processing could be occurred by computer-like transfer and resonance among subunits of cytoskeletal proteins in microtubles.

Hagen, Hameroff, and Tuszynski [2002] have claimed that Tegmark based his calculations on a model which was different from the Orch- Or model and Tegmark's

assumptions should be amended. Subsequently to it, Hameroff and Tuszynski [2004] published a paper that microtuble subunit tublins underwent coherent excitations, which led to the automatic sequence, where quantum coherence superposition was emerged in certain tublins and consciousness was occurred in the brain. But the need to maintain macroscopic quantum cohererence in a warm wet brain is certainly a serious problem for the Penrose-Hameroff model.

HIGH PERFORMANCE COMPUTATION IN THE BRAIN UTILIZING TUNNELING PHOTONS

Contrary to researches mentioned above, we postulate that the human brain is a quantum computer system operated by superluminal evanescent photons.

Figure 8 shows the table of processing speed per CPU of the PowerPC. From which, the ratio of energy loss / processing speed can be roughly estimated as 2.9 J/s/G-Flops.

Power PC	Performance (1CPU)	Watt (1CPU)
PPC603	0.4GFLOPS	6.0W
PPC750	0.6GFLOPS	8.0W
PPC7410	3.2GFLOPS	8.3W
PPC7410	3.7GFLOPS	10.4W
PPC7447	6.4GFLOPS	18.5W

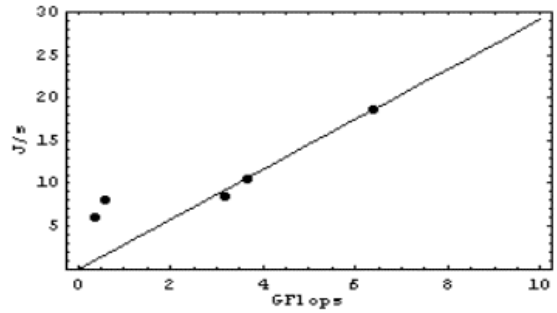


Figure 8. Performance of the Power PC.

Supposing that the computational speed of the human brain is 2×10^{16} operations/sec, the energy of which required for computation by the Power PC can be estimated as 5.8×10^{7} J/s from Figure 8. As human brains consume the energy $500\ kcal$ per day ($=24.2J/s$), the energy ratio of the human brain and the silicon processor becomes $R=4.2\times10^{-7}$, which is similar to the calculation result obtained for QTP gates from Eq.(17). Hence it is seen that the human brain consumes the energy much less than the conventional silicon processors as shown in Figure 3, and it is considered that the human brain is an efficient computational system utilizing superluminal evanescent photons.

From Eqs.(18) and (22), it can be seen that the biological brain has the possibility to perform high efficient computation up to 20 qubits more than conventional silicon processors) with the same energy dissipation when satisfying $\Delta L < L$ as shown in Figure 9 for the infrared light frequency, the wavelength of which is $\lambda = 100\mu m$, where photons would propagate losslessly in a microtubule in the infra-red spectrum region [Hameroff, 2004], if we let $d \approx 15nm$, which is the same order as the extracellular space between the brain cells.

This result suggests that the human brain can perform computation efficiently in the environment at high temperature compared with the conventional processors.

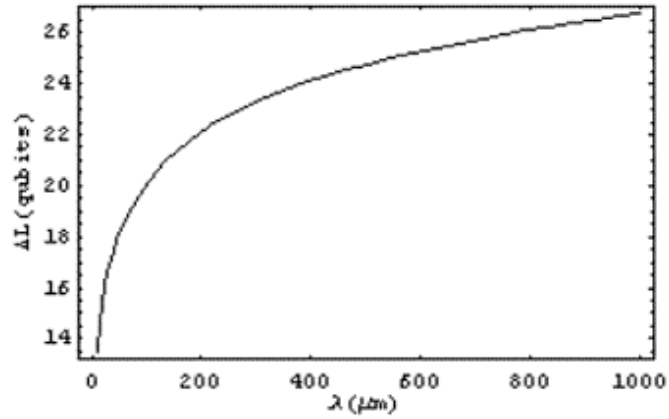


Figure 9. Increase of quantum bits for quantum computation at the infrared spectrum region.

Tegmark [2000] calculated that microtubles in the warm and wet brain would cause decoherence on the order of $\tau \approx 10^{-13}$ sec, which is slower by a factor at least 10^{10} than the time scale of neuron firing, $\tau \approx 10^{-3} \sim 10^{-4}$ sec, from

$$\tau \approx \frac{D^2\sqrt{mk_BT}}{Ngq_e^2} \tag{23}$$

where D is the tublin diameter, m is the mass of an ion, k_B is a Boltzmann constant, $g = 1/4\pi\varepsilon_0$, q_e is an ionic charge and $N = Q/q_e$.

Contrary to his calculation, it can be shown that quantum processor utilizing superluminal evanescent photons can perform computation to satisfy the time scale required for quantum computation in microtubles given as follows;

The decoherence time of the quantum system [Diosi, 2005] can be given by

$$\tau_D = \frac{\hbar^2}{\tau_0}\frac{1}{(\Delta E)^2} \tag{24}$$

where τ_0 is a certain time-scale to measure the strength of time uncertainties.

From Eqs.(20) and (21), the ratio of decoherence times of quantum computation utilizing evanescent (tunnelling) photons and the conventional quantum computation becomes

$$\tau'_D / \tau_D \approx (E_0 / E'_0)^2 = [\beta(\beta - 1)]^2 \qquad (25)$$

When we let $\tau_D = 2.6 \times 10^{-14}$, which is the time required in quantum computation at the room temperature, we have $\tau'_D = \tau_D \times [\beta(\beta-1)]^2 \approx 0.03\,\text{sec}$ for $\lambda = 100\mu m$.

This satisfies the decoherence time, $10^{-5} \sim 10^{-4}\,\text{sec}$, which is required for conducting quantum computation estimated by Hagen, Hameroff and Tsuzynski [2002], and which also satisfies the time scale of neuron firing given by $\tau \approx 10^{-3} \sim 10^{-4}$ sec [Tegmark ,2000]. Conventionally it is supposed that the brain seems far too warm for quantum computation apparently running into the problem of a decoherence time, which would persist for only 10^{-13} seconds at brain temperatures. But recent evidence shows that quantum processes in biological molecules are actually enhanced at higher temperatures [Ouyang and Awschalom,2003].

Thus it is considered that coherent quantum states may be preserved in microtubules at room temperature by superluminal photons, which attain high efficient computation compared with the conventional silicon processors.

SUPER COHERENT ORGANISM VIA TACHYON FIELD

Roger Penrose and Stuart Hameroff have proposed a quantum theory of consciousness to explain that mental processing cannot be wholly mechanical or algorithmic. R.Penrose [1989,1994,1999] proposed his idea in his books on the connection between fundamental physics and human consciousness, in which he argued the possibility of quantum computation in human brains, which is superior to conventional computer systems. It seems to be obvious that a digital computer cannot perform all the functions of the brain [Penrose, 1994].

The quantum computer may be eventually closer to the brain function that the digital computation because classical mechanics cannot fully explain consciousness and quantum mechanical phenomena such as entanglement and superposition may play an important role in the brain's function.

Laszlo [2004] pointed out the puzzle in his book that the living organism is extraordinary coherent with the world around it, dynamically, almost instantly correlated with all other parts, shown as follows;

- The mind of one person appears able to act on the brain and body of another.
- Modern people display a capacity for spontaneous transference of impression and images, especially when they are emotionally close to each other.

This is impossible if the human brain functions similar to the conventional computer systems using silicon processors. The level of coherence exhibited by organism suggests that

quantum type processing take place in them. The living organism, it appears, is in some respects a macroscopic quantum system.

If the brain is functioned by the superluminal particles called a tachyon, this puzzle can be clarified as follows:

From the Klein-Fock-Gordon equation for a scalar field with an imaginary mass $m = i\mu$ given by

$$\left(\frac{\partial^2}{\partial t^2} - \nabla^2 - \mu^2 \right) \psi = 0 \tag{26}$$

we can obtain an elementary solution shown as

$$\psi(x) = \frac{1}{(2\pi)^{3/2}} \exp(ikx) \tag{27}$$

where $\omega = (k^2 - \mu^2)^{1/2}$.

From which, it is seen that tachyons cannot be localized in space from the superposition of solutions given by Feinberg [1967] shown as

$$\psi(x) = \int \phi_k(x) f(k) d^3k \neq \delta^3(x) \tag{28}$$

where $f(k)$ is an arbitrary function, ϕ_k is a set of elementary solutions of the Klein-Fock-Gordon equation with an imaginary mass and $\delta(x)$ is the Dirac's delta function.

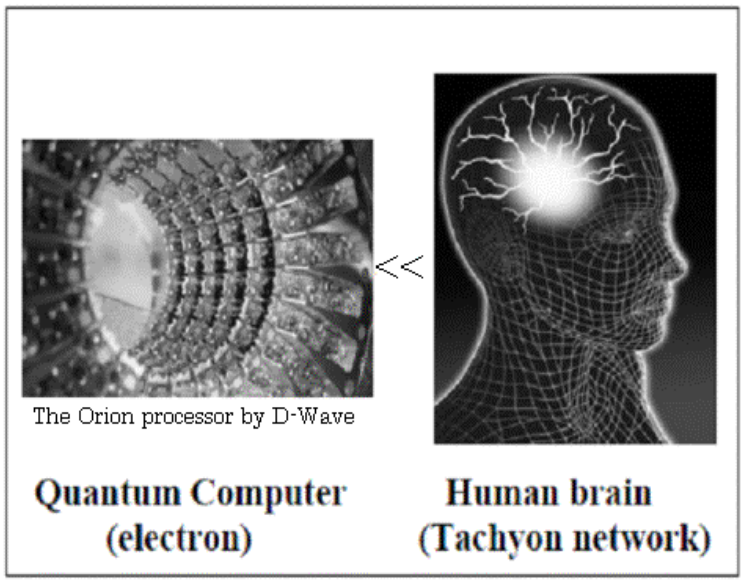


Figure 10. Silicon processor vs. Human brain.

Thus such a superposition cannot be made to vanish outside the sphere of finite radius, but rather influence outside the biological systems. If the brain is a quantum computer system functioned by superluminal photons, it can be entangled with each other via the tachyon field created from the quantum vacuum around it according to the property of non-locality. Thus it is considered that superluminal photons in the brain meditate the long-range physical correlations so that the coherent region inside the brain would behave as inseparable whole, and the human brain can be supposed to function as a macroscopic quantum dynamical system (tachyon network) which can attain much higher performance compared with the computer system utilizing electrons as shown in Figure 10.

The warm and wet inner environment of the brain does not allow any long-time entanglement and superposition of two functional units from the conventional physical mechanism as pointed by the Tegmark's calculation result and thus we must search for other mechanism, for example, via superluminal photons called tachyons.

If this mechanism is true for the brain function, we can obtain the possibility to realize much more efficient computer systems like a human brain by utilizing evanescent photons.

CONCLUSION

On the basis of the theorem that the evanescent photon is a superluminal particle, the possibility of high performance computation in biological systems has been studied. From the theoretical analysis, it is shown that the biological brain has the possibility to achieve large quantum bits computation at the room temperature compared with the conventional processors. Hence it is considered that the human brain can attain high efficient computation compared with the silicon processors.

However it is still remained a question how to determine a qubit in the brain, how it is related to a functioning neuron and how to determine the difference between memory registers and processing units in the brain, and these questions must be clarified by further researches.

REFERENCES

Albrecht-Buehler,G. (1992). Rudimentary form of cellular "vision", *Proc. Natl Acad. Sci. USA*,89(17), 8288-8292.

Benioff,P. (1982). Quantum mechanical models of Turing machines that dissipates no energy, *Physical Review Letters*, Vol.48,No.23,1581-1585.

Bialek, W. and Schweitzer,A. (1987). Quantum noise and the Threshold of Hearing, *Physical Review Letters* 54(7), 725-728.

Brown, J. (1995). Faster than the speed of light, *New Scientist*, Vol.146, 26-30.

Chiao, R.Y., Hickmann, J.M. and Soli, D. Faster-than-Light Effects and Negative Group Delays in Optics and Electronics, and their Applications, 2001. Available from http://arXiv.org/abs/cs/0103014.

Diosi,L. (2005). Instrict Time Uncertainties and Dechoherence: Comparison of 4 models, *Brazilian Journal of Physics*, Vol.35, No.2A, 260-265.

Del Giudise,E., Doglia,S. and Milani,M.(1986). Electromagnetic field and Spontaneous Symmetry Breaking Biological Matter, *Nuclear Physics*, B257, 185-199.

Faber,J., Portugal, R. and Rosa,L.P. (2006). Information processing in brain microtubles, *Biosystems*, 83, 1-9.

Feinberg,G. (1967). Possibility of faster-than–light particles, *physical review,* Vol.159, No.5, 1089-1100.

Feynman,R.P. (1986). Quantum Mechanical Computers, *Foundation of Physics*, Vol.16,No.6, 507-531.

Gea-Banacloche,J. (2005). Fundamental limit to quantum computation: the energy cost of precise quantum logic, Mathematical Sciences (No.508, pp.47-57), Tokyo: Saiensu Co.Ltd, also Available from: http://comp.uark.edu/~jgeabana/fundamentalv3.pdf.

Hagan,S., Hameroff,S.R. and J.A.Tuszynski,J.A. (2002). Quantum computation in brain microtubles: Decoherence and biological feasibility, *Physical Review* E, Vol.65, 061901-1-061001-11.

Hameroff,S.R. (1982). Information Processing in microtubles, *J. Theor. Biol*. 98, 549-561.

Hameroff,S.R. (2004). A new theory of the origin of cancer: quantum coherent entanglement, centrioles, mitosis, and differentiation, *Biosystems,* Vol.77, Issues 1-3, 119-136.

Hameroff,S.R. and Penrose,R. (1996). Conscious Events as Orchestrared Space-Time entanglement, Selections, *Journal of Consciousness Studies,* Vol.3, 36-53.

Hameroff,S.R. and Tuszynski,J. (2004). Quantum states in proteins and protein assemblies: The essence of life?, SPIE Conference, Grand Canary island, May.

Haroche,S. and Raimond,J-M. (1996). Quantum Computing: Dream or Nightmare? , *Physics Today*, Vol.49, No.8,51-52.

Jibu,M.,Hagan,S.,Hameroff,S.R.,Pribram,K.H. and Yasue,K. (1994). Quantum optical coherence in cytoskeletal microtubles: implications for brain function, *Biosystems,* 32 (3),195-209.

Jibu,M.,Pribram,K.H. and Yasue,K. (1996). From Conscious Experience to Memory Storage and Retrieval: The Role of Quantum Brain Dynamics and Boson Condensation of Evanescent Photons, *Int. J. Mod. Phys. B*, Vol.10.No.13and14,1753-1754.

Jibu,M. and Yasue,K. (1997a). What is mind?-Quantum Field Theory of Evanescent Photons in Brain as Quantum Theory of Consciousness, *Informatica* 21,471-490.

Jibu,M., Yasue,K and Hagan,S. (1997b). Evanescent (tunneling) photon and cellular vision, *Biosystems,* 42,65-73.

Laszlo,E., Science and the Akarshic Field, An Integral Theory of Everything, Innner Traditions, Rochester, Vermont, 2004.

Lloyd, S. (2000). Ultimate physical limits to computation, *Nature,* Vol.406, 1047-1054.

Margolus,N.and Levitin,L.B. (1998). The maximum speed of dynamical evolution, *Physica D,* 120,188-195.

Musha,T. (2006). A study on the possibility of high performance computation using quantum tunneling photons*, Int. J. Simulation and Process Modeling*, Vol.2, Nos.1/2,63-66.

Musha, T. (2009). Possibility of high performance quantum computation by superluminal evanescent photons in living systems, *BioSystems*, 96, 242-245.

Ouyang,M. and Awschalom,D.D. (2003). Coherent spin transfer between molecularly bridged quantum dots, *Science* 301, 1074-1078.

Park,M. and Park,Y. (1996). On the foundation of the relativistic dynamics with the tachyon, *Nuovo Cimento*, Vol.111B,N.11, 1333-1368.

Penrose, R., The Emperor's New Mind, Oxford University Press, Oxford,1989.

Penrose, R., Shadows of the Mind, A Search for the Missing Science of Consciousness, Oxford University Press, Oxford, 1994.

Penrose, R., The Large, the Small and the Human mind, Cambridge University Press, Cambridge, 1999.

Recami,E. (2001). A bird's-eye view of the experimental status-of-the-art for superluminal motions, *Found. of Phys.* 31,1119-1135.

Satinover, J., The Quantum Brain, John Wiley and Sons, Inc., New York, 2001.

Smolyyaninov, I.I, Zayats, A.V., Gungor, A. and Davis, C.C. Single-Photon Tunneling, 2004. Available from: http://arXiv.org/PS-cache/cond-mat/pdf/0110/0110252.pdf.

Steinberg, A.M., Kwait, P.G. and Chiao, R.Y. (1993). Measurement of the single-photon tunneling time, *Physical Review Lett*ers, Vol.71, No.5, 708-710.

Tegmark,M. (2000). Importance of quantum coherence in brain processes, *Physical Review*, E61,4194-4206.

Unruh,W.G. (1995). Maintaining coherence in quantum computers, *Physical Review* A,Vol.51,No.2, 992-997.

Vitiello,G. (1995). Dissipation and memory capacity in the quantum brain model, *Int. J. Mod. Phys. B,* Vol.9, Issue.8, 973-989.

Wang, L.J., Kuzmich, A. and Dogariu, A. (2000). Gain-assisted superluminal light propagation, *Nature*, Vol.406, 277-279.

Whipple, C.T. Nano Switching in Optical Near-Field, Eye on Technology, oe magazine. 2002. Available from: http://www.oemagazine.com/from TheMagazine/mar02/pdf/ eyeontech.pdf.

In: Computer Science Research and Technology. Volume 2 ISBN 978-1-61122-073-5
Editor: Janice P. Bauer

Chapter 5

DESIGN OF METRO HUBS FOR OPTICAL METRO NETWORKS

Yiu-Wing Leung[1]
Department of Computer Science, Hong Kong Baptist University,
Kowloon Tong, Hong Kong

ABSTRACT

Nowadays many local networks provide high-speed access to end users (e.g., Gigabit Ethernet and optical access networks). Accordingly, each metro network should have a large bandwidth in order to transport the traffic between the local networks and the wide area backbone network. To fulfill this requirement, an *optical metro network* uses optical fibers to connect the local networks to the backbone network via a *metro hub*. In this chapter, we describe our recent design of metro hubs called *multi-passive-combiner hubs* [1-2]. A multi-passive-combiner hub is composed of passive combiners and demultiplexers. Within the hub, wavelength channels are assigned to the input-output pairs to fulfill the traffic requirements while avoiding wavelength conflict. The multi-passive-combiner hub has three major advantages: i) it has low cost and high reliability because it only uses passive optical components, ii) it can efficiently transport non-uniform metro traffic via non-uniform channel assignment within the hub, and iii) it can easily be scaled up to provide more channels for the ever-increasing traffic load.

1. INTRODUCTION

The Internet can be viewed as a three-level hierarchy: local networks, metro networks and backbone networks [3]. Nowadays many local networks provide high-speed access to end users (such as Gigabit Ethernet and optical access networks [4-5]), and a backbone network can provide very large bandwidth by using optical fibers [6-12]. Accordingly, a metro network should have a large bandwidth to transport the traffic among many local networks

[1] Email: ywleung@comp.hkbu.edu.hk.

and the backbone network. A *metro wavelength-division-multiplexing (WDM) network* adopts the WDM technology [13-14] to provide multiple wavelength channels for this purpose.

Figure 1 shows one of the popular architectures for metro WDM networks. In this architecture, a *metro hub* connects the local networks to the backbone network via optical fibers. There are N nodes where node i is connected to the local network in region i ($1 \le i \le N-1$) and node N is connected to the wide area backbone network. Each node can send data to any other nodes through the metro hub. It is desirable that the metro hub only uses passive optical components for low cost and high reliability. In the literature, two types of hubs were proposed and they are described in the following.

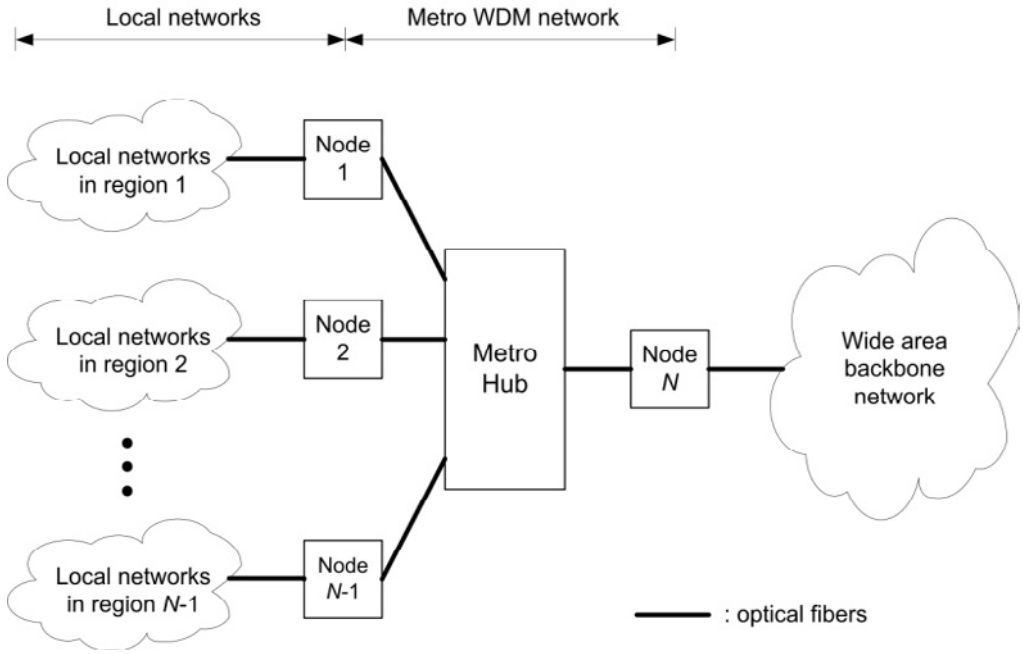


Figure 1. Metro WDM network in which each node carries the traffic of many users.

The first type of metro hubs is called *passive-star-coupler hub* [15-17]. This metro hub uses a *passive star coupler* to provide W wavelength channels to its input-output pairs [15-17], where the passive star coupler is a passive optical component and W is the number of wavelength channels in a fiber. Figure 2 shows an example of a passive-star-coupler hub with W=4. The hub can provide W wavelength channels to the node pairs. If the number of nodes N is equal to the number of wavelength channels W, then node i can be assigned a dedicated wavelength channel at λ_i for reception (this is shown in the example in Figure 2). If node j wants to send data to node i, node j transmits at wavelength λ_i. On the other hand, if N is not equal to W, the N nodes can share the W wavelength channels through a multiple access control protocol.

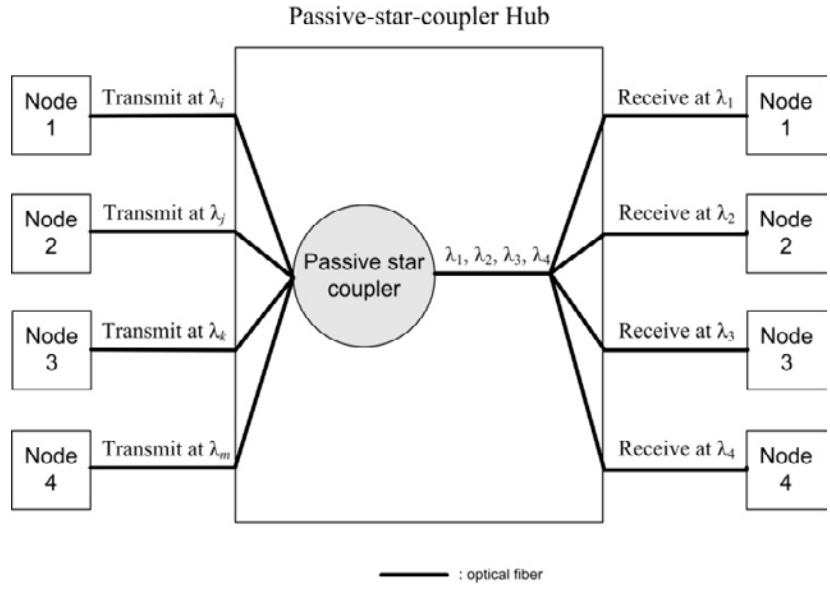


Figure 2. Example of a passive-star-coupler hub [15-17].

The second type of metro hubs is called AWG hub [3, 18-22]. This metro hub uses an *arrayed waveguide grating (AWG)* to provide one wavelength channel to every input-output pair, where the AWG is a passive optical component. Figure 3 shows an example of an AWG hub. This hub can provide more wavelength channels than a passive-star-coupler hub [3, 18-22]. Because of this advantage, AWG hubs have been studied extensively in the literature [3, 18-22] and implemented as field trials in Europe [18] and Japan [19].

In our recent studies [1-2], we observe two important requirements for metro networks:

1. Metro networks should be able to transport non-uniform traffic efficiently. It is because non-uniform traffic is common in many practical scenarios. At the network edge, the node (e.g., node *N* in Figure 1) has heavier traffic because it transports the traffic between all the local networks and the backbone network. In the network core, different regions may have different number of users (e.g., in some cities, different regions have very different population density) and hence they may have different traffic volume.
2. Metro networks should have good scalability (i.e., the networks can easily be scaled up to provide more channels for the ever-increasing traffic load). It is because bandwidth demand has been increasing for two reasons: i) there are more and more users with high-speed access (e.g., provided by Gigabit Ethernet or optical access networks), and ii) there are more and more bandwidth-demanding applications (e.g., multimedia entertainment services, content distribution, virtual private networks, grid computing, etc.). In fact, the Internet traffic was doubling every six months in the past few years [23], and it is envisioned that the bandwidth demand will grow exponentially in the coming 20 years [24].

The existing passive-star-coupler hub and the AWG hub cannot fulfill the above requirements. In particular, the passive-star-coupler hub can only provide W channels which may not be enough for the ever-increasing traffic load, and the AWG hub provides uniform channel allocation (specifically, one channel to every input-output pair) and hence it cannot efficiently transport non-uniform traffic.

In our recent studies [1-2], we propose a new design of metro hubs called *multi-passive-combiner hubs*. This new design can provide the same communication functions as the existing hubs while it can also fulfill the above two requirements (i.e., the multi-passive-combiner hub can efficiently transport non-uniform traffic and it can easily be scaled up to provide more channels to fulfill the ever-increasing bandwidth demand).

In this chapter, we summarize our research work on multi-passive-combiner hubs [1-2]. In Section 2, we describe the multi-passive-combiner hubs. In Section 3, we determine the channel requirements for the input-output pairs based on the traffic load and patterns. In Section 4, we assign wavelength channels to the input-output pairs to fulfill the channel requirements while avoiding wavelength conflict. In Section 5, we present numerical results to demonstrate the effectiveness of the multi-passive-combiner hubs. In Section 6, we conclude this chapter.

2. DESIGN OF MULTI-PASSIVE-COMBINER HUBS

We consider the network model shown in Figure 1. The metro network has N nodes. In each local region, a node serves the end terminals in this region and grooms their traffic into one or more wavelength channels. The nodes are connected to the metro hub through optical fibers. The multi-passive-combiner hub is shown in Figure 4. Within the hub, each incoming fiber is connected to a demultiplexer, the demultiplexers are connected to the passive combiners based on a channel assignment (to be determined in Section 4), and each passive combiner is connected to an outgoing fiber. This hub only involves passive components (namely, demultiplexers and passive combiners) and hence it has low cost and high reliability. In the following, we explain how the multi-passive-combiner hub can provide the same communication functions as the AWG hub (which is a well-regarded design in the literature) as well as other desirable functions.

The multi-passive-combiner hub can provide the same communication as the AWG hub as follows. Let the inputs/outputs of the hub be numbered from 1. An AWG hub connects wavelength channel λ_i from input j to output $(j+i-2) \bmod W+1$ (e.g., see Figure 3). A multi-passive-combiner hub can provide the same communication function as follows: it uses W demultiplexers and W passive combiners, connects wavelength channel λ_i from input j to passive combiner $(j+i-2) \bmod W+1$, and connects passive combiner j to output j. Figure 4 shows a multi-passive-combiner hub which provides the same communication function as the AWG hub shown in Figure 3.

The multi-passive-combiner hub can efficiently transport non-uniform metro traffic as follows. It can allocate different number of channels to different input-output pairs based on their traffic requirements (the details are given in Section 3). In other words, it can realize non-uniform channel allocation within the hub to best fit the non-uniform traffic. On the other hand, the AWG hub provides one wavelength channel to each input-output pair (e.g., see

Figure 3) and this uniform channel allocation is not well suited for non-uniform traffic. Figure 5(a) and (b) show two examples in which the multi-passive-combiner hubs realize non-uniform channel allocation. In these examples, the multi-passive-combiner hubs use the same number of channels as the AWG hub shown in Figure 3 but they can better utilize these channels for non-uniform traffic.

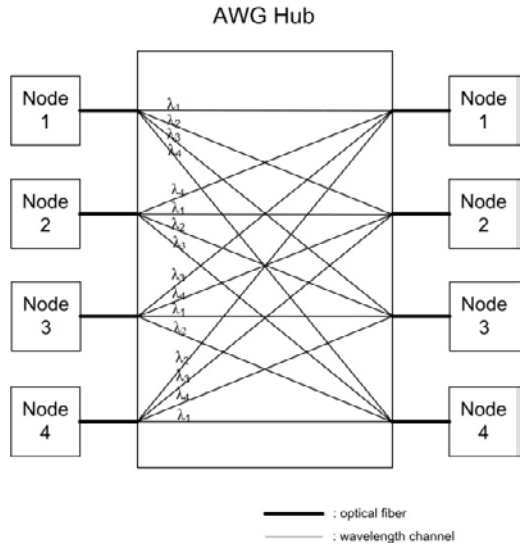


Figure 3. Example of an AWG hub [3, 18-22]. The AWG hub can provide more channels than the passive-star-coupler hub shown in Figure 2.

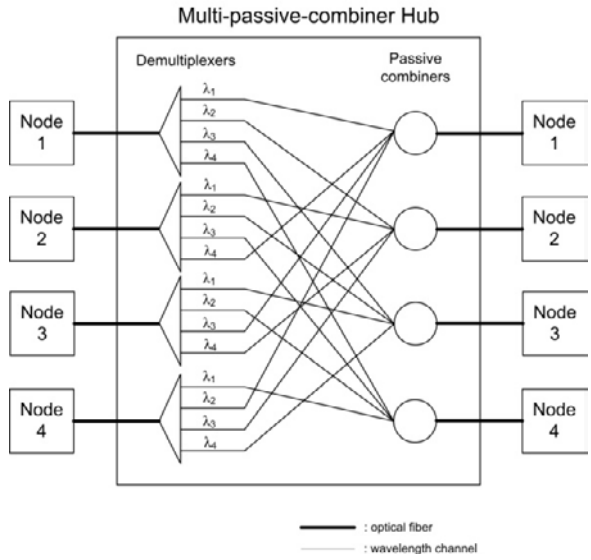
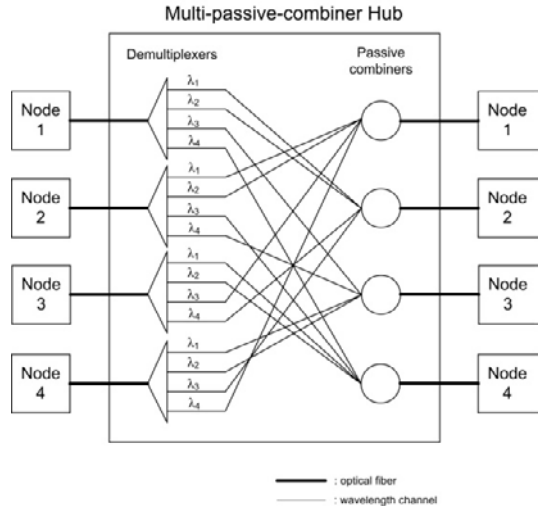


Figure 4. Multi-passive-combiner hub proposed by the author in [1-2]. In this example, the hub provides the same communication function as the AWG hub shown in Figure 3 (i.e., provides one wavelength channel to each input-output pair).

The multi-passive-combiner hub can easily be scaled up to provide more channels so that it can sustain to handle the ever-increasing traffic load. This can be achieved by using more demultiplexers and passive combiners within the hub. The cost involved is low because demultiplexers and passive combiners are low-cost passive components. Figure 6 shows an example in which the multi-passive-combiner hub provides more channels than the one shown in Figure 4 while it realizes non-uniform channel allocation for non-uniform traffic.

(a)



(b)

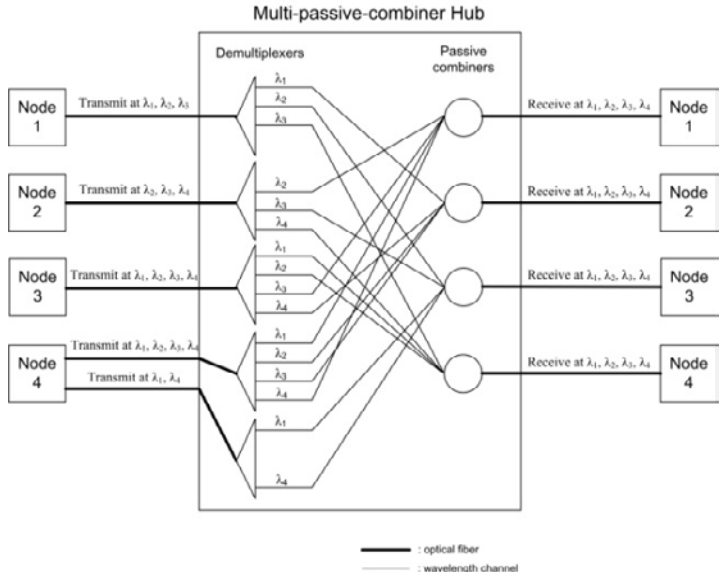


Figure 5. Multi-passive-combiner hub can realize non-uniform channel allocation for non-uniform traffic. (a). First example of non-uniform channel allocation. (b). Second example of non-uniform channel allocation.

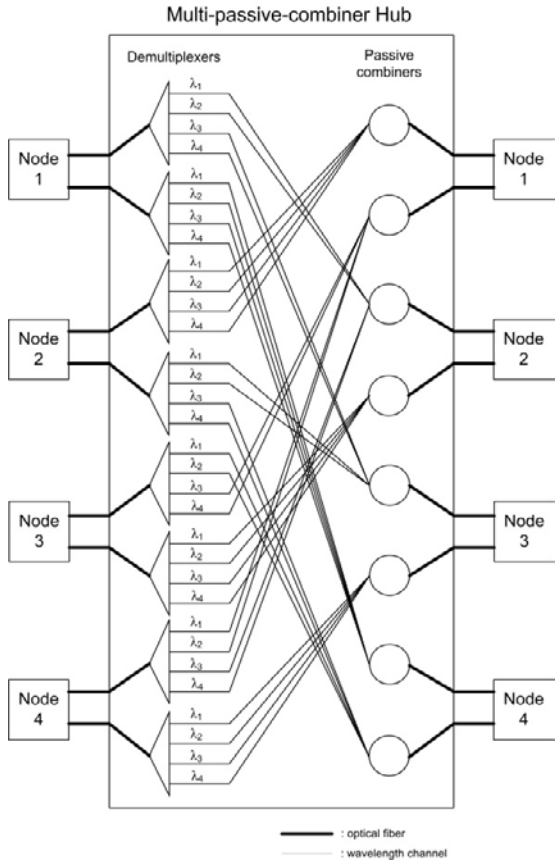


Figure 6. Multi-passive-combiner hub can support more incoming/outgoing fibers to provide more channels for heavy traffic while it can realize non-uniform channel allocation for non-uniform traffic.

3. CHANNEL REQUIREMENTS

Consider the network model shown in Figure 1. Node i ($1 \le i \le N-1$) carries the traffic of all users in servicing region i, while node N carries the traffic of all users. In this section, we determine the number of channels required by each node pair based on the traffic load and pattern.

Let $\mathbf{D} = [d_{ij}]_{N \times N}$ be a *channel requirement matrix* where d_{ij} is the number of channels required from node i to node j. To determine d_{ij}, we consider the communication subsystem from input i to output j of the multi-passive-combiner hub. We model this communication subsystem as a queueing system with d_{ij} servers. Based on any given traffic model and statistics, we determine the mean packet delay $\overline{Delay_{ij}}$ in this queueing system as a function of d_{ij} using the existing methodology [25]. Then we determine the smallest d_{ij} such that the mean packet delay $\overline{Delay_{ij}}$ is smaller than a given requirement $Delay^*$ (e.g., $Delay^*$ is given to be 1 ms such that the mean delay $\overline{Delay_{ij}}$ is at most 1 ms). This can be done by using the

bisection method because the mean packet delay $\overline{Delay}_{ij}$ is a monotonic decreasing function of the number of channels d_{ij}. The details of determining the channel requirement d_{ij} for all $1 \le i, j \le N$ are given as follows.

1. Select d_{ij}^{low} and d_{ij}^{up} such that $\overline{Delay}_{ij}\Big|_{d_{ij}=d_{ij}^{low}} > Delay^*$ and $\overline{Delay}_{ij}\Big|_{d_{ij}=d_{ij}^{up}} \le Delay^*$.
2. If $(d_{ij}^{up} - d_{ij}^{low}) = 1$ then $d_{ij} = d_{ij}^{up}$ and STOP.
3. While $(d_{ij}^{up} - d_{ij}^{low}) > 1$ do the following
4. Begin
5. $d_{ij}^{mid} = \left\lfloor \dfrac{d_{ij}^{low} + d_{ij}^{up}}{2} \right\rfloor$
6. If $\overline{Delay}_{ij}\Big|_{d_{ij}=d_{ij}^{mid}} = Delay^*$, then $d_{ij} = d_{ij}^{mid}$ and STOP.
7. Else if $\overline{Delay}_{ij}\Big|_{d_{ij}=d_{ij}^{mid}} > Delay^*$, then $d_{ij}^{low} = d_{ij}^{mid}$.
8. Else if $\overline{Delay}_{ij}\Big|_{d_{ij}=d_{ij}^{mid}} < Delay^*$, then $d_{ij}^{up} = d_{ij}^{mid}$.
9. End_while
10. $d_{ij} = d_{ij}^{up}$ and STOP.

We now consider a specific traffic model based on the above framework. Let the packet arrival process from node *i* to node *j* be Poisson with mean packet rate $\alpha_{i,j}$, and the packet size be exponentially distributed with mean packet transmission time $1/\beta$. The communication subsystem from node *i* to node *j* can be modeled as an $M/M/d_{ij}$ queueing system. The mean packet delay $\overline{Delay}_{ij}$ can be found to be [25]:

$$\overline{Delay}_{ij} = \frac{(d_{ij}\rho_{ij})^{d_{ij}} C}{\beta d_{ij} d_{ij}!(1-\rho_{ij})^2} + \frac{1}{\beta} \tag{1}$$

where $\rho_{ij} = \alpha_{ij} / d_{ij}\beta$ and

$$C = \frac{1}{\sum_{k=0}^{d_{ij}-1} \dfrac{(d_{ij}\rho_{ij})^k}{k!} + \dfrac{(d_{ij}\rho_{ij})^{d_{ij}}}{d_{ij}!(1-\rho_{ij})}} \tag{2}$$

We determine the smallest d_{ij} such that the mean packet delay $\overline{Delay}_{ij}$ is smaller than the given requirement $Delay^*$. The following is an example. Let the mean packet arrival rate (in terms of number of packets per second) be given by

$$[\alpha_{ij}]_{N\times N} = \begin{bmatrix} 0 & 1.0\times10^6 & 1.5\times10^6 & 2.0\times10^6 \\ 1.5\times10^6 & 0 & 2.0\times10^6 & 3.0\times10^6 \\ 1.5\times10^6 & 2.0\times10^6 & 0 & 3.0\times10^6 \\ 3.0\times10^6 & 3.5\times10^6 & 4.0\times10^6 & 0 \end{bmatrix} \quad (3)$$

and the mean packet transmission time be 1 μs. If the mean packet delay cannot be larger than $Delay^* = 1.5\mu s$, then the channel requirement matrix can be found to be:

$$\mathbf{D} = \begin{bmatrix} 0 & 2 & 3 & 3 \\ 3 & 0 & 3 & 5 \\ 3 & 3 & 0 & 5 \\ 5 & 5 & 6 & 0 \end{bmatrix} \quad (4)$$

If the mean packet delay requirement is relaxed to $Delay^* = 5.0\mu s$, then the channel requirement matrix becomes:

$$\mathbf{D} = \begin{bmatrix} 0 & 2 & 2 & 3 \\ 2 & 0 & 3 & 4 \\ 2 & 3 & 0 & 4 \\ 4 & 4 & 5 & 0 \end{bmatrix} \quad (5)$$

In other words, when the mean packet delay requirement is looser, fewer channels are needed.

4. CHANNEL ASSIGNMENT

After determining the number of channels d_{ij} required from node *i* to node *j*, it is necessary to assign d_{ij} wavelength channels to this node pair. In this assignment, if two or more channels at the same wavelength are connected to the same passive combiner within the multi-passive-combiner hub, then there is *wavelength conflict* and the signals transmitted in these channels interfere with each other. Therefore, it is important to avoid *wavelength conflict* such that only the channels at distinct wavelengths are connected to each passive combiner. For example, wavelength conflict is avoided in the multi-passive-combiner hub shown in Figure 7. In this section, we define the channel assignment problem for the multi-passive-combiner hubs and propose an efficient algorithm for channel assignment.

4.1. Model and Problem Definition

We consider the network model shown in Figure 1. The metro network has N nodes where nodes 1, 2, …, N–1 are connected to N–1 respective local regions and node N is connected to the wide area backbone network. In each local region, a node serves all the end terminals in this region and grooms their traffic into one or more wavelength channels. The nodes are connected to the multi-passive-combiner hub through optical fibers.

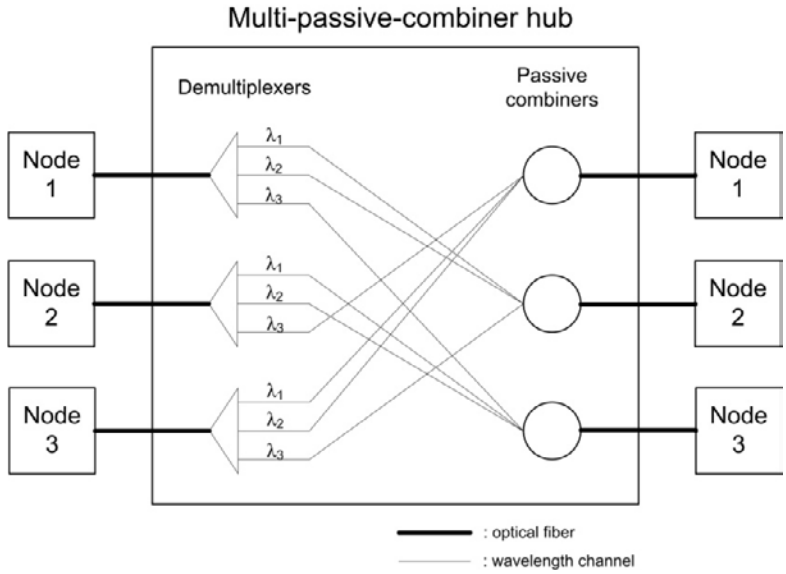


Figure 7. In this example, wavelength conflict is avoided because only the channels at distinct wavelengths are connected to each passive combiner.

The channel assignment problem is defined as follows: assign d_{ij} wavelength channels from node i to node j within the metro hub for all $1 \le i, j \le N$ (i.e., to fulfill the channel requirements determined in Section 3), such that only the channels at distinct wavelengths are connected to each passive combiner (i.e., to avoid wavelength conflict within the multi-passive-combiner hub).

4.2. Channel Assignment Method

Our main idea for channel assignment is as follows. We first transform the channel requirement matrix **D** to another representation such that the channel assignment problem becomes a sequence of known combinatorial problems called *distinct representatives problems* [26]. We solve the latter problems by applying the existing distinct representatives algorithm, and then transform the resulting solutions to get the channel assignment.

Before explaining the details of channel assignment, we briefly review the concept of distinct representatives [26]. Given the sets Ω_1, Ω_2, $\cdots$, Ω_n, we select one item (called *representative*) from each set such that the selected representatives are different from each

other. Mathematically, the distinct representatives of Ω_1, Ω_2, $\cdots$, Ω_n are z_1, z_2, $\cdots$, z_n where $z_i \in \Omega_i$ and $z_i \neq z_j$ for all $1 \le i \le n$ and $j \neq i$. An efficient polynomial-time algorithm for finding distinct representatives is given in [26].

Our channel assignment method consists of three main steps and they are described in the following.

Step 1. Transformation

Node *i* requires $r_i \equiv \sum_{j=1}^{N} d_{ij}$ channels to the hub and $c_i \equiv \sum_{j=1}^{N} d_{ji}$ channels from the hub, so it requires $\lceil r_i/W \rceil$ fibers to the hub and $\lceil c_i/W \rceil$ fibers from the hub. Overall, the hub has $\sum_{i=1}^{N} \lceil r_i/W \rceil$ incoming fibers and $\sum_{j=1}^{N} \lceil c_j/W \rceil$ outgoing fibers, so it has $\sum_{i=1}^{N} \lceil r_i/W \rceil$ demultiplexers and $\sum_{j=1}^{N} \lceil c_j/W \rceil$ passive combiners. We first transform $\mathbf{D}$ to a matrix $\mathbf{D}'$ which specifies the number of channels required from every demultiplexer to every passive combiner. This is done as follows: for all $1 \le i \le N$, if $r_i > W$, we divide row *i* into $\lceil r_i/W \rceil$ rows such that each of these $\lceil r_i/W \rceil$ rows has a row sum of at most *W* and the sum of the j^{th} element of these $\lceil r_i/W \rceil$ rows is equal to d_{ij} for all $1 \le j \le N$; for all $1 \le j \le N$, if $c_j > W$, we divide column *j* into $\lceil c_j/W \rceil$ columns such that each of these $\lceil c_j/W \rceil$ columns has a column sum of at most *W* and the sum of the i^{th} element of these $\lceil c_j/W \rceil$ columns is equal to d_{ij} for all $1 \le i \le \sum_{k=1}^{N} \lceil r_k/W \rceil$. Then we add dummy row(s) or column(s) to $\mathbf{D}'$ such that the resulting $\mathbf{D}'$ has *M* rows and *M* columns where $M = \max(\sum_{i=1}^{N} \lceil r_i/W \rceil, \sum_{j=1}^{N} \lceil c_j/W \rceil)$, and add dummy values to some elements in $\mathbf{D}'$ such that all row sums and column sums are equal to *W*.

Step 2. Solving the Transformed Problem

For the $M \times M$ matrix $\mathbf{D}'$ with equal row sum and column sum, there exists *M* distinct representatives such that each distinct representative appears in one distinct row and one distinct column [26]. We apply this property to get a conflict-free channel assignment at each distinct wavelength based on $\mathbf{D}'$. Specifically, we form *M* sets Ω_1, Ω_2, $\cdots$, Ω_M where Ω_i contains the column numbers of the non-zero elements in row *i* of $\mathbf{D}'$, and execute the distinct representatives algorithm [26] to determine *M* distinct representatives z_1, z_2, $\cdots$, z_M of these *M* sets. We use these distinct representatives to form a tentative conflict-free assignment at λ_1: assign a wavelength channel at λ_1 to connect demultiplexer *i* to passive combiner z_i for all $1 \le i \le M$. We record this tentative assignment at λ_1 by the *assignment set* $\mathrm{A}(\lambda_1) = \{(1, z_1), (2, z_2), \cdots, (M, z_M)\}$. Then we update $\mathbf{D}'$ by deducting $d_{iz_i}{}'$ by 1 for all $1 \le i \le M$. The resulting $\mathbf{D}'$ still has equal row sum and column sum. We repeat the above

to get the assignment sets $\mathrm{A}(\lambda_2)$, $\mathrm{A}(\lambda_3)$, $\cdots$, $\mathrm{A}(\lambda_W)$ for the channels at λ_2, λ_3, $\cdots$, λ_W respectively.

Step 3. Channel Assignment

We remove the dummy items added in step 1 from the corresponding ones in $\mathrm{A}(\lambda_1)$, $\mathrm{A}(\lambda_2)$, $\cdots$, $\mathrm{A}(\lambda_W)$. Then we perform channel assignment at λ_k based on $\mathrm{A}(\lambda_k)$ for all $1 \le k \le W$ as follows: for every $(i, z_i) \in \mathrm{A}(\lambda_k)$, assign a wavelength channel at λ_k to connect demultiplexer *i* to passive combiner z_i.

In the above channel assignment method, the most time-consuming step is to solve a sequence of *W* distinct representative problems. Each of these problems can be solved in $O(M^3)$ steps [26]. Therefore, our channel assignment method has a time complexity of $O(WM^3)$.

4.3. Numerical Example

We consider a multi-passive-combiner hub with four nodes and the following channel requirement matrix:

$$\mathbf{D} = \begin{bmatrix} 0 & 3 & 1 & 6 \\ 2 & 0 & 1 & 7 \\ 1 & 2 & 0 & 2 \\ 7 & 5 & 3 & 0 \end{bmatrix} \tag{6}$$

Let $W = 5$. After executing step 1 of the channel assignment method, the channel requirement matrix **D** is transformed to the following representation:

$$\mathbf{D}' = \begin{bmatrix} 0 & 0 & 3 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 1 & 0 \\ 2 & 0 & 0 & 0 & 1 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 3 \\ 1 & 0 & 2 & 0 & 0 & 0 & 0 & 2 \\ 2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 3 & 0 & 0 & 0 \end{bmatrix} \tag{7}$$

Correspondingly, the four nodes have 2, 2, 1, 3 fibers to the hub respectively and 2, 2, 1, 3 fibers from the hub respectively. After executing steps 2 and 3, we get the following assignment sets:

$$\begin{aligned} \mathrm{A}(\lambda_1) &= \{(1,5),(2,6),(3,7),(4,8),(5,3),(6,1),(7,2),(8,4)\} \\ \mathrm{A}(\lambda_2) &= \{(1,3),(2,6),(3,7),(4,8),(5,1),(6,2),(7,4),(8,5)\} \\ \mathrm{A}(\lambda_3) &= \{(1,3),(2,6),(3,5),(4,7),(5,8),(6,1),(7,2),(8,4)\} \end{aligned} \tag{8}$$

$$A(\lambda_4) = \{(1,3),(2,6),(3,1),(4,7),(5,8),(6,2),(7,4),(8,5)\}$$
$$A(\lambda_5) = \{(1,6),(2,7),(3,1),(4,8),(5,3),(6,2),(7,4),(8,5)\}$$

We assign the wavelength channels at λ_1 based on $A(\lambda_1)$ as follows: assign a channel at λ_1 to connect demultiplexer 1 to passive combiner 5, assign another channel at λ_1 to connect demultiplexer 2 to passive combiner 6, etc. Similarly, we assign the wavelength channels at λ_2, λ_3, λ_4, λ_5 based on $A(\lambda_2)$, $A(\lambda_3)$, $A(\lambda_4)$, $A(\lambda_5)$ respectively. The details of channel assignment are shown in Figure 8.

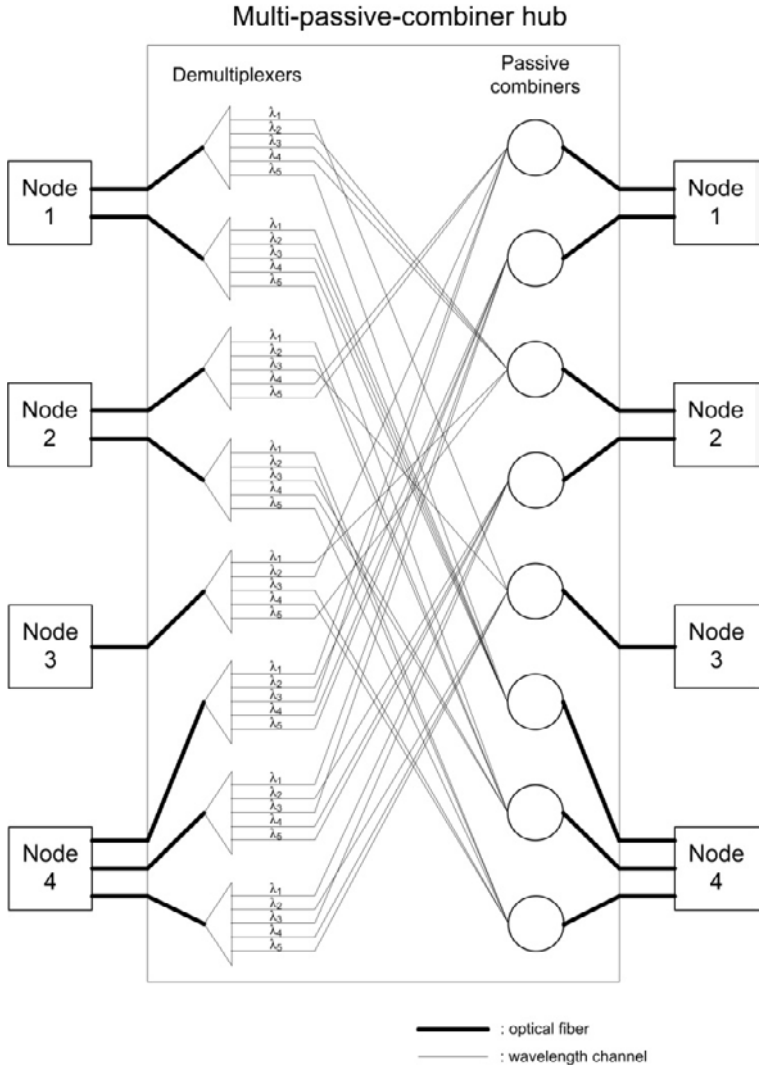


Figure 8. An example of channel assignment for the multi-passive-combiner hub. This channel assignment fulfills the channel requirements given by equation (6) while it has no wavelength conflict because only the channels at distinct wavelengths are connected to each passive combiner.

5. NUMERICAL RESULTS

In this section, we compare the performance of the multi-passive-combiner hub with the AWG hub under different traffic patterns and loads (we remind that the AWG hub has the best performance in the literature). We consider a metro WDM network which connects three local networks to the backbone network. Node 4 has a heavier traffic to/from other nodes because it transports the traffic between all the three local networks and the backbone network (e.g., see Figure 1 with N=4). Let the packet arrival process be Poisson and the packet arrival rate α_{ij} from node i to node j be:

$$\alpha_{ij} = \begin{cases} 0 & i = j \\ \alpha & i \neq j \neq 4 \\ f \times \alpha & i \neq j \text{ and } (i = 4 \text{ or } j = 4) \end{cases} \tag{9}$$

where the parameters α and f determine the traffic load and traffic pattern respectively. If f=1, the traffic is uniform; if f is larger, the traffic is more non-uniform. Let the packet size be exponentially distributed, the mean packet transmission time be 1 μs, the packet delay requirement $Delay^*$ for the multi-passive-combiner hub be 2 μs, and W=4.

Figure 9(a) shows the mean packet delay under uniform traffic. We see that the multi-passive-combiner hub gives smaller mean delay especially when the traffic load is heavy. It is because the multi-passive-combiner hub can be suitably scaled to provide enough channels to fit the traffic requirements. More specifically, when the traffic load increases, the mean delay of the multi-passive-combiner hub increases until it reaches the given delay requirement $Delay^* = 2\mu s$. At this point, when the traffic load further increases, one additional passive combiner will be used to avoid exceeding $Delay^* = 2\mu s$ so that the mean delay decreases. As a result, the mean packet delay is always smaller than the given requirement $Delay^* = 2\mu$s.

(a) (b)

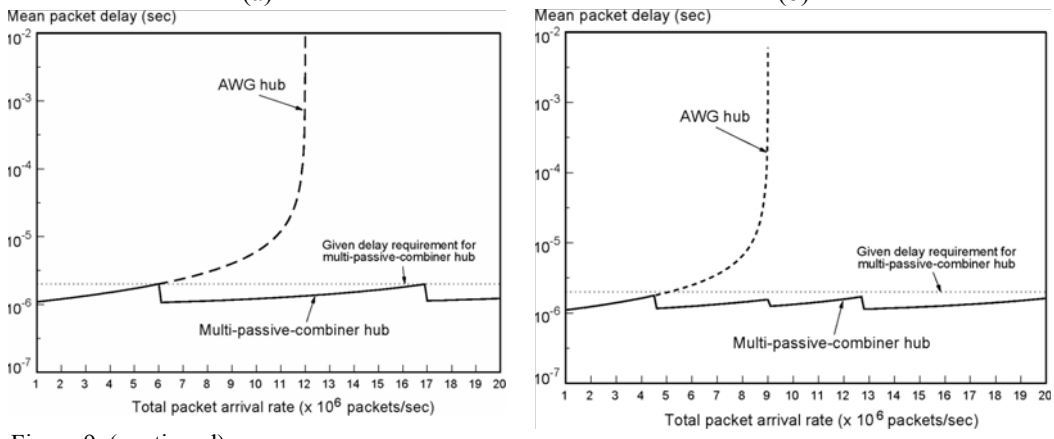


Figure 9. (continued)

(c)

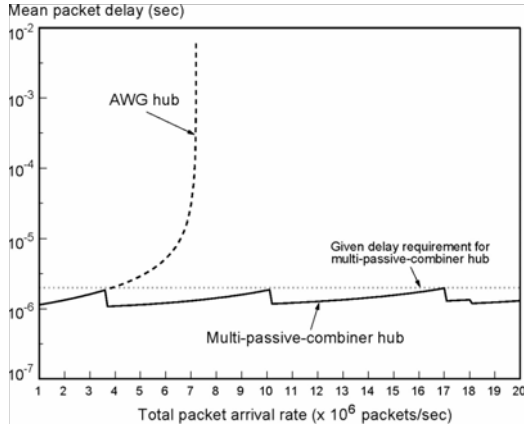


Figure 9. Mean packet delay in multi-passive-combiner hub and AWG hub.(a) Uniform traffic. (b) Non-uniform traffic (*f*=2). (c) Non-uniform traffic (*f*=5, i.e., traffic is more non-uniform than that in Figure. 9(b)).

Figure 9(b) shows the mean packet delay under non-uniform traffic at $f = 2$. For non-uniform traffic, the AWH hub has longer mean delay because of its inherent uniform channel allocation within the hub, while the multi-passive-combiner hub has smaller mean delay (which does not exceed the given requirement $Delay^* = 2\mu s$) because it can have non-uniform channel assignment to fit the non-uniform traffic pattern. Figure 9(c) shows that when the traffic is more non-uniform with $f = 5$, the multi-passive-combiner hub has even better performance than the AWG hub. These results show that the multi-passive-combiner hub is more effective when the traffic load is heavy or the traffic pattern is non-uniform.

CONCLUSION

We described our recent design of multi-passive-combiner hubs for metro WDM networks. A multi-passive-combiner hub is only composed of passive optical components (namely, passive combiners and demultiplexers) and it assigns wavelength channels to the input-output pairs based on the traffic load and patterns while avoiding wavelength conflict. As a result, the multi-passive-combiner hub offers three major advantages: i) it has low cost and high reliability because it only uses passive optical components, ii) it can efficiently transport non-uniform metro traffic via non-uniform channel assignment within the hub, and iii) it can easily be scaled up to provide more channels for the ever-increasing traffic load.

REFERENCES

[1] Y. W. Leung, "Multipassive-star-coupler hub for metro WDM networks," *IEEE Photonic Technology Letters,* vol. 18, no. 15, pp. 1621-1623, August 2006.

[2] Y. W. Leung, "Channel assignment for a metro hub under nonuniform traffic," *IEEE Photonic Technology Letters,* vol. 20, no. 21, pp. 1787-1789, November 2008.

[3] M. Scheutzow, M. Maier, M. Reisslein, and A. Wolisz, "Wavelength reuse for efficient packet-switched transport in an AWG-based metro WDM network," *IEEE Journal of Lightwave Technology,* vol. 21, no. 6, pp. 1435-1455, June 2003.

[4] H. Shinohara, "Broadband access in Japan: rapidly growing FTTH market," *IEEE Communications Magazine*, vol. 43, no. 9, pp. 72-78, September 2005.

[5] F. Effenberger, D. Cleary, O. Haran, R. D. Li, M. Oron, and T. Pfeiffer, "An introduction to PON technologies," *IEEE Communications Magazine*, vol. 45, no. 3, pp. S17-S25, March 2007.

[6] T. K. C. Chan and Y. W. Leung, "Homogeneous and heterogeneous subnetting for constructing all-optical multi-fiber networks," *IEEE Journal of Lightwave Technology*, vol. 27, no. 13, pp. 2271-2281, July 2009.

[7] T. K. C. Chan, E. W. M. Wong, and Y. W. Leung, "High-capacity time-domain wavelength interleaved networks," *IEEE Journal of Lightwave Technology*, vol. 27, no. 17, pp. 3948-3958, September 2009.

[8] Y. W. Leung, "Lightpath concentrators for all-optical networks," *IEEE Journal of Lightwave Technology*, vol. 24, no. 9, pp. 3259-3267, September 2006.

[9] T. K. C. Chan, E. W. M. Wong, and Y. W. Leung, "Shared-by-wavelength-switches: a node architecture using small optical switches and shared wavelength converters," *IEEE Photonics Technology Letters*, vol. 18, no. 12, pp. 1335-1337, June 2006.

[10] Y. W. Leung, G. Xiao, and K. W. Hung, "Design of node configuration for all-optical multi-fiber networks," *IEEE Transactions on Communications*, vol. 50, no. 1, pp. 135-145, January 2002.

[11] G. Xiao, Y. W. Leung, and K. W. Hung, "Two-stage cut saturation algorithm for designing all-optical networks," *IEEE Transactions on Communications*, vol. 49, no. 6, pp. 1102-1115, June 2001.

[12] G. Xiao and Y. W. Leung, "Algorithms for allocating wavelength converters in all-optical networks," *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, pp. 545-557, August 1999.

[13] R. Ramaswami and K. N. Sivarajan, *Optical Networks: A Practical Perspective*, 2nd ed., Morgan Kaufmann, 2002.

[14] M. Maier, *Optical Switching Networks*, Cambridge University Press, 2008.

[15] B. Kannan, S. Fotedar, and M. Gerla, "A two level optical star WDM metropolitan area network," *Proc. IEEE GLOBECOM*, November 1994.

[16] J. M. Senior, M. R. Handley, and M. S. Leeson, "Developments in wavelength division multiple access networks," *IEEE Communications Magazine*, pp. 28-36, December 1998.

[17] J. M. H. Elmirghani and H. T. Mouftah, "Technologies and architectures for scalable dynamic dense WDM networks," *IEEE Communication Magazine*, pp. 58-66, February 2000.

[18] A. Bianco, E. Leonardi, M. Mellia, and F. Neri, "Network controller design for SONATA: a large-scale all-optical passive network," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 2017-2028, October 2000.

[19] K. Noguchi, Y. Koike, H. Tanobe, K. Harada, and M. Matsuoka, "Field trial of full-mesh WDM network (AWG-STAR) in metropolitan/local area," *IEEE Journal of Lightwave Technology*, vol. 22, no. 2, pp. 329-336, February 2004.

[20] C. J. Chae, H. Park, and Y. Park, "Hybrid optical star coupler suitable for wavelength reuse," *IEEE Photonic Technology Letters*, vol. 10, no. 2, pp. 279-281, February 1998.

[21] H. S. Yang, M. Herzog, M. Maier, and M. Reisslein, "Metro WDM networks: performance comparison of slotted ring and AWG star networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 8, pp. 1460-1473, October 2004.

[22] N. Kamiyama, "A large-scale AWG-based single-hop WDM network using couplers with collision avoidance," *IEEE Journal of Lightwave Technology*, vol. 23, no. 7, pp. 2194-2205, July 2005.

[23] A. K. Somani, M. Mina, and L. Li, "On trading wavelengths with fibers: a cost-performance based study," *IEEE/ACM Transactions on Networking*, vol. 12, no. 5, pp. 944-950, October 2004.

[24] E. B. Desurvire, "Capacity demand and technology challenges for lightwave systems in the next two decades (invited paper)," *IEEE Journal of Lightwave Technology*, vol. 24, no. 12, pp. 4697-4710, December 2006.

[25] L. Kleinrock, *Queueing Systems*, Vol. 1: Theory, Wiley, 1975.

[26] M. Hall, *Combinatorial Theory*, Chapter 5: Distinct Representatives, 2nd ed., John Wiley and Sons, 1986.

In: Computer Science Research and Technology. Volume 2 ISBN 978-1-61122-073-5
Editor: Janice P. Bauer

Chapter 6

A BLUETOOTH SENSOR NETWORK BASED ON THE IEEE 1451 STANDARD A SENSOR NETWORK SOLUTION TO EVALUATE THE WELLNESS OF THE PASSENGER AND IMPROVE THE SAFETY IN CARS

Jesus Murgoitio Larrauri[1], Beñat Arejita Larrinaga[2], Maider Larburu Lopez[3], and Javier Sanchez Cubillo[4]
Robotiker-Tecnalia, Parque Tecnológico, Edificio 202,
Zamudio (Bizkaia), Spain

ABSTRACT

The use of sensors is highly extended in a lot of different environments and applications. Each situation needs a different solution and for that reason the use of a scalable and easily manageable sensor network is a must while applications are getting more and more complex. In a lot of cases the perfect solution is the one based on a wireless sensor network; it gives flexibility, ease of management of the system and expandability. But in order to facilitate interoperability between different sensor manufacturers and to give a transparent and independent interface, the use of a standard is mandatory. This standard system is provided by the IEEE 1451 family of standard protocols. In this project a Bluetooth based sensor network has been implemented using the IEEE 1451 family of standard protocols. The goal of this network is to help on data acquisition from a number of sensors within a car, in order to monitor the wellness of the passengers and improve the safety and comfort.

[1] murgoiti@robotiker.es.
[2] barejita@robotiker.es.
[3] mlarburur@robotiker.es.
[4] jsanchez@robotiker.es.

Keywords: Sensor network, IEEE 1451, Bluetooth, NCAP, STIM, TEDS, plug and play

1. INTRODUCTION

The Bluetooth sensor network that is presented in this paper has been developed for an automotive environment in order to increase the safety and the passengers comfort and to monitor their wellness. For this purpose several sensors installed within a car have been controlled using an IEEE 1451 standard based sensor network.

The sensor network has been implemented within a Medea+ project called Caring Cars, with satisfactory evaluation.

As a result, some electronic boards have been developed based on the HCS08 microcontroller family, which are provided with Bluetooth capabilities. The firmware of these devices has been implemented using the APIs and data structures defined in the IEEE 1451.0 standard. In parallel, some software libraries and their respective APIs have been created in order to control these devices from a Linux based PC.

2. CARING CARS

The main goal of the Caring Cars project was to increase car safety by enabling wellness applications in an automotive environment.

The main target was in-car safety and wellness to address the huge indirect costs of transportation in the EU. Reports of the European Environment Agency estimate the indirect costs of transportation at about 8% of GDP, a substantial part of which is caused by accidents. Each year an estimated 127 thousand people are killed and about 2.4 million are injured on roads in Europe.

It was the main goal of this project to address these costs by turning the car into a safer and better environment.

To achieve it, the project designed an open automotive infrastructure, the basis of which were formed by a sensor network in cooperation with a car gateway. This sensor network consists of the sensors already available in vehicles augmented with new sensors. The Car gateway manages and coordinates in-car devices and establishes a connection with the external world signalling for enriched information exchange. In this way it will be possible to improve car safety and thus reduce the costs of transportation.

By adding external communication to the infrastructure envisioned by the project it will also become possible to use the same infrastructure to support health care applications.

This project had partners from both industry and academia to form a well balanced consortium that has experience in providing car manufacturers and manufacturing cars whereas it contained technical expertise on the relevant technical fields. The consortium used this experience and knowledge to lift car safety to the next level.

3. IEEE 1451

The IEEE 1451 family of standard protocols is aimed to give a standard set of commands and data structures to facilitate the management of sensor networks and transducers themselves. It gives to the application layer not only a transparent interface to handle sensors and actuators but also a communication interface that is independent to the communication protocol that is used in the physical layer.

The standard defines two main entities, the Network Capable Application Processor (NCAP called device) and the Smart Transducer Interface Module (STIM called device).

In order to achieve this, the standard family is divided into different subfamilies: IEEE 1451.0, IEEE 1451.1, IEEE 1451.2, IEEE 1451.3, IEEE 1451.4, IEEE 1451.5, IEEE 1451.6 and IEEE 1451.7.

In the following figure a diagram of the entire family is shown.

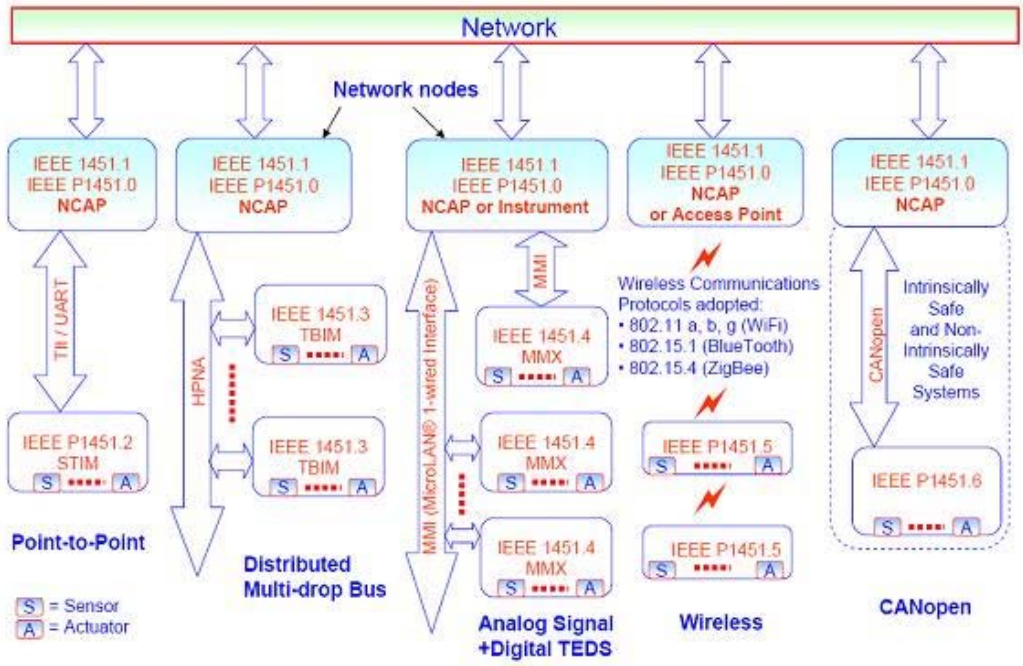


Figure 1. IEEE 1451 family of protocols.

One key feature of the IEEE 1451 is the Transducer Electronic Data Sheet, TEDS. This kind of file is used to help to implement plug and play features for the management of the transducers and the transducer network itself.

In this project the 1451.0 and 1451.5 subfamilies have been used. The first one defines all the data structures, commands, TEDS and communication, transducer services and HTTP access APIs and the second one defines the interface for IEEE 1451 wireless communications (802.11.x, 802.15.1, 802.15.4).

4. BLUETOOTH SENSOR NETWORK

The sensor network that has been implemented, consist of three main elements. In the highest level of the hierarchy there is the car-gateway which is in charge of collecting all the data from the sensor network and store it in the data base.

One step below in the architectural hierarchy is the sensor manager, which is focused on collecting all the data from the sensor nodes using a Bluetooth communication and then sends the collected information to the car-gateway via Ethernet.

Finally, there is one more element in a lower level of the architectural hierarchy, the sensor node. It works as an interface to the sensors themselves. It is able to control eight different sensors and send the gathered information to the sensor manager via Bluetooth using the IEEE1451.0 TransducerServices API and ModuleCommunication API.

4.1. Network Architecture

The network architecture that has been used is illustrated in figure 2.

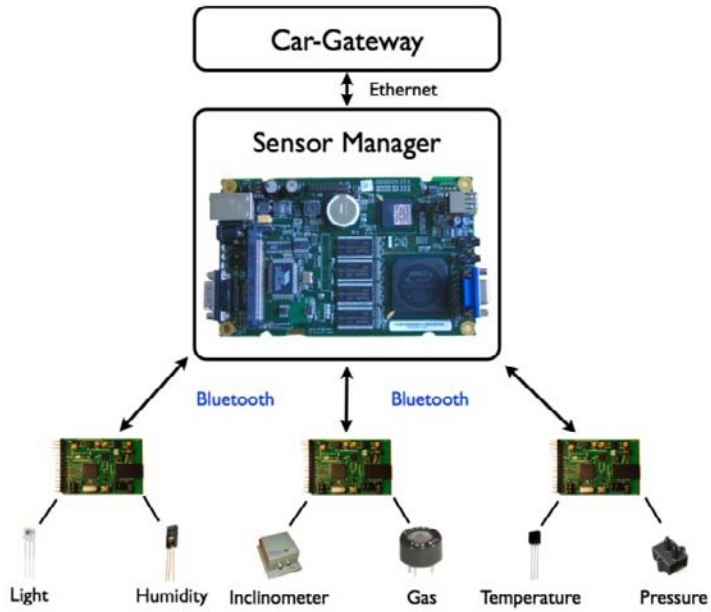


Figure 2. General architecture of the sensor network.

Figure 2 shows the main elements of the system: the car-gateway, the sensor manager and the sensor nodes.

For the Caring Cars project, seven different transducers have been used: gas detection, temperature, pressure, inclination, humidity, light and a special ECG signal sensor too.

All these transducers have been distributed within a car, making three different groups each one controlled by a sensor node (STIM). The data acquired by the nodes is sent to the sensor manager using a previously established Bluetooth connection. The communication parameters and the connection itself are established by the sensor manager.

4.2. Car-Gateway

The Car-Gateway is the top element of the architecture and it is a Linux based Car-PC. Its responsibility is to run the main application, storing the information sent by the Sensor Manager via Ethernet and evaluating the information of the sensors in the system, and finally taking the right decision in order to help increasing car safety and passengers comfort.

4.3. Sensor Manager, NCAP

The Sensor Manager is a Linux based mini-PC (alix 3c3). It uses the created libraries and APIs to manage the sensors and the communication with the sensor nodes in a transparent way, as it is defined in the IEEE 1451.0 and IEEE 1451.5 standards.

The following table summarizes the functions that have been implemented in the first version of this solution:

Table 1. Implemented functions

Function name	Description
TedsManager_readTeds	Reads TEDS files
TimDiscovery_reportTims	Reports available STIMs
TimDiscovery_reportChannels	Reports available channels in a STIM
TransducerAccess_open	Opens the access of the specified channel
TransducerAccess_close	Closes the access of the specified channel
TransducerAccess_startStream	Starts the data stream from a specified channel
TransducerAccess_cancel	Stops the data stream of a specified channel
Comm_init	Initializes the communication parameters and establishes the Bluetooth connection
Comm_shutdown	Shuts down a previously stablished Bluetooth connection
NetComm_open	Opens the communication interface of the IEEE 1451 layer
NetComm_close	Closes the communication interface of the IEEE 1451 layer
NetComm_readMsg	Reads a message that is in the Bluetooth layer from the IEEE 1451 layer
NetComm_writeMsg	Writes a message from the IEEE 1451 layer in the Bluetooth layer
NetComm_discoverDestinations	Discovery of the destination of the message
P2PComm_read	Reading of a Point-to-Point communication
P2PComm_write	Writing of a Point-to-Point communitcation
P2PComm_flush	Flushes the buffers of a Point-to-Point communication
P2PComm_readSize	Reads the size of a message of a Point-to-Point communication for memory allocation
P2PReceive_notifyMsg	Notifies an incoming message to the IEEE 1451 layer

4.4. Sensor Nodes, STIM

The sensor nodes are 8 bit microcontroller based solutions. They are able to control eight different transducers and they have a Bluetooth chip, the LMX9838. It communicates with the microcontroller using a serial communication interface, and it has implemented the Bluetooth protocol stack until the SDP layer.

The software in the microcontroller sends the information via serial lines in messages defined by the Bluetooth chip manufacturer (national semiconductor).

The established Bluetooth connection uses the RFCOMM protocol to emulate a serial point to point communication between the sensor nodes and the sensor manager using radio signals.

4.5. Sensors and TEDS

The IEEE 1451.0 standard defines the so called Transducer Electronic Data Sheet or TEDS. These files are data structures divided in perfectly defined different fields. The information within these fields describes the sensor itself.

Using the information of these files the application layer can easily know every parameter of the sensor that is controlling, for instance calibration information, sensor's measuring ranges or maximum and minimum working temperature of the sensor.

One of the most useful characteristic of these files is that they can store the information for the conversion from the sensor's output voltage value to its corresponding physical units. These characteristic makes easier the management of the transducers giving to the application a transparent interface to work with them, regardless of the physical characteristics of the transducer in question.

An example of a calibration TEDS code for the LM35 temperature sensor is shown here:

```
XDCR12_CAL_TEDS_SIZE, /* Total Length */
0x03, /* TEDS identification header */
0x04, /* Length of TEDS */
0x00, /* Family */
0x05, /* Class / Access code */
0x01, /* Version of the standard */
0x01, /* Tuple length */
0x0A, /* Last calibration date */
0x08, /* Length of field */
0x4A, 0x43, 0xD7, 0x40, /* Jun 25, 2009, 20:00:00 */
0x00, 0x00, 0x00, 0x00, /* See above */
0x0B, /* Calibration interval */
0x08, /* Length of field */
0x01, 0xE1, 0x33, 0x80, /* 1 year */
0x00, 0x00, 0x00, 0x00, /* See above */
0x0C, /* SIConvrt */
0x0C, /* Length of field */
0x1E, /* SISlope */
```

```
0x04, /* Length of field */
0x42, 0xC8, 0x00, 0x00, /* 100 */
0x1F, /* Intrcpt */
0x04, /* Length of field */
0x00, 0x00, 0x00, 0x00, /* 0 */
0x14, /* Linear conversion */
0x0A, /* Length of field */
0x33, /* Set of coefficients */
0x08, /* Length of field */
0x3A, 0xA0, 0x00, 0x00, /* Slope: 0.001220703125 */
0x00, 0x00, 0x00, 0x00, /* Intercept: 0 */
XDCR12_CAL_TEDS_CSUM /* Checksum */
```

The strangest sensor used in this project is the ECG signal sensor. It is not very usual to obtain the ECG signal in a car environment but in the other hand this signal gives a lot of information about the wellness and comfort of the passengers.

To obtain this signal three conductive material stripes have been added to the surface of the steering wheel. These stripes are put in ways that are able to obtain the necessary signals to obtain the so wanted ECG signal.

The use of such a rare sensor in automotive environment has been facilitated thanks to the sensor network and the IEEE 1451 family of standard protocols.

4.6. Results

As a result of this project, some IEEE 1451 standard compatible electronic boards have been manufactured which are able to communicate using the very well known Bluetooth protocol.

Figure 3. Demonstrator panel.

An open standard API and its respective libraries have been developed in order to be able to control the sensor nodes in a transparent way not only for the management of the sensors, but also for the communications used in the physical layer.

Finally a functional sensor network has been implemented in a car environment satisfactorily controlling a number of sensors between which the ECG sensor has been the strangest but at the same time the most valuable sensor to monitor the wellness and comfort of the passenger.

The previous photo shows the demonstrator panel fabricated in order to explain the main feature provided by this technology, which is the easy way to interchange transducers connected to different channels in any sensor node (STIM), only downloading the proper TEDS from the NCAP to the corresponding STIM.

CONCLUSIONS

The goals of this sensor network have been satisfactorily achieved. A functional sensor network has been implemented based on Bluetooth communications and IEEE 1451 family of standard protocols.

The system that has been built in this project has helped to implement an application that is able to control and manage a number of sensors in a transparent way, regardless of the physical communication used in the lower layers and regardless of the physical characteristics of the sensors.

The use of Bluetooth communications has helped to easily implement the sensor network in a car environment due to the wireless characteristic of the communication. The bandwidth and data transfer rates provided by Bluetooth has been far enough to satisfactorily measure all the sensors at real time.

From another point of view the use of IEEE 1451 family of standard protocols has been very useful. The data structures, the transducer access interface and the transparent communication interface provided by the standard family have been the key features for the success of the entire sensor network.

Last but not least, an open software library and its respective API have been developed to access the sensor network using a Linux based computer.

REFERENCES

Bluetooth Special Interest Group. *Specification of the Bluetooth system*, URL www.bluetooth.com.

Lee, K.B. Schneeman, R.D. *Distributed Measurement and Control Based on the IEEE 1451 Smart Transducer Interface Standards, IEEE transaction on Instrumentation and Measurement.* Volume 49, Issue 3, Pages 621-627.

Song, E.Y. Lee, K.B. 2008. STWS: *A Unified Web Service for IEEE 1451 Smart Transducers, IEEE transaction on Instrumentation and Measurement.* Volume 57, Issue 8, Pages 1749-1756.

The Institute of Electrical and Electronics Engineers, 2007. *IEEE Standard for a Smart Transducer Interface for Sensors and Actuators—Common Functions, Communication Protocols, and Transducer Electronic Data Sheet (TEDS) Formats.*

The Institute of Electrical and Electronics Engineers, 2007. *IEEE Standard for a Smart Transducer Interface for Sensors and Actuators—Wireless Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats.*

The Institute of Electrical and Electronics Engineers. *IEEE Std 802.15.1-2005, Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs),* 14 June 2005.

The Institute of Electrical and Electronics Engineers. *IEEE Std 802.15.4-2006, Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs),* 8 September 2006.

In: Computer Science Research and Technology. Volume 2 ISBN 978-1-61122-073-5
Editor: Janice P. Bauer

Chapter 7

SABOTAGE-TOLERANT JOB SCHEDULING FOR RELIABLE VOLUNTEER COMPUTING

Kan Watanabe* *and Masaru Fukushi* †
Graduate School of Information Sciences, Tohoku University, Japan

Abstract

Volunteer computing (VC) is a type of Internet based parallel computing paradigm, which allows any participants on the Internet to contribute their idle computing resources. By exploiting the resources that can be found in millions of Internet computers, VC makes it possible to build a very large and high performance global computing systems with a very low cost. However, VCs still have the problem of guaranteeing computational correctness, due to the inherent unreliability of volunteer participants. In this chapter, we fully study the sabotage-tolerance performance of several sabotage-tolerance methods including popular voting method used in real VCs, and reveal their drawbacks in terms of computation time and error rate. Then, based on the conclusion from the comparison study, we propose a new job scheduling method for credibility-based voting to efficiently attain the sabotage-tolerance of VCs, i.e. the mathematical expectation of computational error rate is bounded by a given error rate specified as reliability requirement. The key idea for efficient job scheduling is to choose jobs to be executed prior to others considering the important feature that the necessary redundancy for each job may change with time. Extensive simulation has shown that the proposed job scheduling method reduces the overall computation time of VCs, while guaranteeing the reliability requirement.

Key Words: sabotage-tolerance, credibility-based mechanisms, reliability, job scheduling

1. Introduction

1.1. Volunteer Computing

Volunteer computing (VC) is a variation on the idea of metacomputing - using a network of many separate computers as if they were one large parallel machine [1].

*E-mail address: can@ecei.tohoku.ac.jp
†E-mail address: mfukushi@ecei.tohoku.ac.jp

Metacomputing systems today primarily take the form of networks of workstations (NOWs) [3], that allow people to pool together existing and mostly idle workstations in their own local or institution-wide networks and use them to do parallel processing without having to purchase an expensive supercomputer [4]. More recent metacomputing projects have started going beyond the bounds of individual institutions, and have started addressing the problems of security, reliability, and scalability in nation-wide and global networks. These projects include Legion [5], Globus [6], NetSolve [7], and Ninf [8]. Since then, a lot of metacomputing research have been focused on creating "grid computing" [9] environment, which aim at providing seamless access to computing resources around the country and the world.

VC is a form of metacomputing that focuses on maximizing the ease with which people can have their machines be part of a metacomputer [1]. While other forms of metacomputing such as grid computing seek to make "using" compute servers on the network easy, "setting up" these compute servers is currently not as easy and requires helps of experts. Thus, metacomputing systems still have complex setup requirements that prevent ordinary users from participating in these systems. The idea behind VC is to eliminate these setup requirements, and make it so easy to join a computing network that anyone, even a casual computer user without any technical knowledge, can do it. By allowing casual users to contribute their computers' processing power in this way, VC makes it possible to harness the power of thousands, or even millions, of computers in very little time.

In recent years, there are two big reasons why VC should be focused on [2]. The first reason is that the prices of personal computers have dropped drastically. The price drops have made it possible for many households to have one or more computers. In addition to family computers, many college students have their own computers and the large number of colleges have many computers in labs and libraries for student use. A lot of these computers are connected to the Internet and many computers remain turned on 24 hours a day, only being used for part of the day and sitting idle during the rest of the day. These computers have the potential to be used for VC during those idle periods.

The second is that there is a trend of putting more than one core on a CPU so that personal computers are now coming with multiple processing cores instead of a single one. While personal computers with these multi-core CPUs are now truly capable of doing multiple tasks at the same time, people still have trouble doing multiple tasks at once. Therefore, we expect that there will be an increase in the number of CPU cycles that are used by the system' s idle process. Other people than the computers' owners could potentially use these CPU cycles without noticeably affecting the computers user' s experience. These changes have the potential to really increase the amount of CPU cycles donated to VC projects.

1.2. Problems and Motivations

While there has been a rapidly growing interest in VC as a ultrahigh performance computing environment, VC still has a mandatory issue for reliable computing. Different from the grid computing system which shares managed reliable computers within or among organizations, computing resources of VC system are owned and managed by volunteer participants. Those resources may behave erratically due to hardware/software failures or virus infection, or may behave maliciously to falsify the computation, each of which results in

sabotage to the computation. It is reported in [27] that a large fraction of participants (about 35 %) actually returned at least one erroneous result in a real VC.

Against the sabotaging, some sabotage-tolerance mechanisms are proposed for reliable computation. However, (i) the sabotage tolerant performance and the drawbacks of those mechanisms have been hidden in real VC systems since they are proposed and evaluated based on unrealistic VC models; (ii) There is no dynamic scheduling method although the sabotage-tolerance performance is closely related to job scheduling in that redundant jobs for sabotage-tolerance are allocated to volunteers in a job scheduling procedure. Therefore, major VC frameworks still provide the simplest sabotage-tolerance mechanism based on a static job scheduling, which significantly degrades the performance instead of ensuring the reliability of computation.

To solve this problem, we propose a dynamic job scheduling method for sabotage-tolerance mechanisms and evaluate it based on practical VC model which considers workers' dynamic activities. The key idea is to introduce novel metrics, i.e. expected-credibility and the expected number of results, for dynamic job scheduling. These metrics take account of active (running) jobs and show actual condition of redundant computation, which works to avoid excess job executions and performance degradation. Different from the static round robin method used in [23], the proposed method selects a proper job dynamically based on these metrics.

The remainder of this chapter is organized as follows: Section 2 provides some background knowledge of VCs and describes the basic and more practical VC models. Section 3 introduces existing sabotage-tolerance mechanisms and compares their performance in VC models described in Section 2. Section 4 proposes a dynamic job scheduling method for sabotage-tolerance mechanisms and demonstrates the performance. Section 5 concludes this chapter with a summary and an indication of our future work direction.

2. VC Platform

In this section, we will provide some background knowledge of VCs which will help to understand the main body of this chapter. We will first introduce major VC projects which have run successfully or is currently running on the Internet (Sec. 2.1), and the remaining sections cover VC framework (Sec. 2.2), VC model (Sec. 2.3), and existing problem and motivation of this chapter (Sec. 2.4).

2.1. VC Projects

Large scale VC projects began in the 1990's with The Great Internet Mersenne Prime Search (GIMPS) [13] and Distributed.net [14]. These projects allow people to solve many parallel problems that were previously computationally infeasible. The application areas of VC include medicine, science, chemistry, and mathematics. More than 20 VC projects are currently running, including the well-known SETI@home project, which searches for evidence of extraterrestrial life [11, 12]. The purpose of these projects are searching for cures for diseases, looking for evidence of extraterrestrial intelligence, finding Mersenne prime numbers, and so on. Below is the list of some famous VC projects.

- Great Internet Mersenne Prime Search
  Great Internet Mersenne Prime Search (GIMPS) [13] is the oldest of the major VC projects, stated in 1996. In the ten years since this project has been running, the computers participating in it have discovered thirteen previously unknown Mersenne primes, the most recently discovered one being the second largest known prime number (the 47th known Mersenne prime, $2^{42,643,801} - 1$, on April 12th, 2009).

- Distributed.net
  Distributed.net [14], the organization that started the second largest VC project, began in 1997. Distributed.net used their VC projects to win several cryptographic challenges, including the CS-Cipher Challenge, and four challenges sponsored by RSA Laboratories.

- SETI@Home
  SETI@home [11, 12] is probably the most well known VC project. The project, conceived in 1995 by David Gedye and started in May 1999, searches for extraterrestrial life by processing the signals collected by radio telescopes. In addition to just scanning the data for signs of extraterrestrial life, the SETI@home client program displays a screen saver showing information about the signals it is currently processing. As of April 30, 2010, over two million years of CPU time (72,962,101,899 credit [19]) had been contributed to this project [18].

2.2. VC Frameworks

VC projects are developed based on the client-server architectural model. The VC projects require both hardware and software on both client and server sides. Project's sponsors provide server hardware and both server and client software, while volunteers supply client hardware as shown in Fig.1.

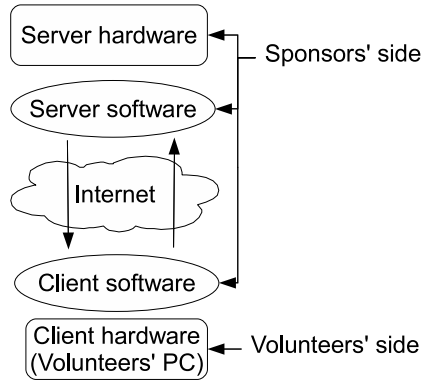


Figure 1. Provisions of VC components .

The server has many responsibilities in VC projects. The server's first responsibility is managing the infrastructure for the project. This consists of hosting a web page where volunteers can learn about the project and download the client software. The server must

also manage and store the information about volunteers, including how much work their clients have accomplished. The next function the server must perform is generating the jobs (also known as workunits) which the clients will perform. When the server receives a request for a job from a client, the server should assign such job to the client that is likely to be completed by an associated deadline.

In contrast to the server portion of VC projects, the client portion has few responsibilities. Client software typically performs the following. When the client runs for the first time, a set of benchmarks is run on the computer the client resides on. These benchmarks indicate the computational power the client is likely to be able to contribute in some time interval. Then the client requests jobs from the server and receives a set of filenames, corresponding to jobs, to download from the server. The client downloads the files and then begins processing them, one at a time.

Because the portions of the server software and the client software are common in almost all VC projects, it is possible to develop a set of tools, or a framework, for creating VC projects. A framework can simplify the work of creating a VC project by automating many of the tasks required to create a project and providing templates or skeletons for programs that must be written by the project sponsor. There are several important VC frameworks. The most influential frameworks are BOINC, Xtremweb and Bayanihan, which were developed in academia.

- Berkeley Open Infrastructure for Network Computing
  Berkeley Open Infrastructure for Network Computing (BOINC) [17] is one of the dominant frameworks currently used to build VC projects. In many VC projects, such as SETI@home [11, 12], Einstein@home [15] and Rosetta@home [16], BOINC is used for their frameworks. BOINC allows a person with a single computer (to act as the server) to create a VC project without writing all the networking code for a client and server by providing the scripts and tools to set up everything except the actual processing code, which the project creator supplies.

- Xtremweb
  Xtremweb [10], developed at the University of Paris Sud, is another framework for creating VC projects. Like BOINC, Xtremweb provides tools to assist in the creation of VC projects. In contrast to BOINC, Xtremweb only allows each client to retrieve one task at a time; when a client completes its assigned task and returns the result, it can request another task.

- Bayanihan
  Bayanihan [21, 22] aims to take VC even further by developing web-based VC systems where programmers can write platform-independent parallel applications in Java and post them on the Web as applets so that volunteers need only a web browser and a few mouse clicks to join a computation. To participate in Bayanihan, volunteers visited the web page with their web browser. The browser then automatically downloaded and ran a Java applet. A variety of benefits can be found for such web-based VC systems, ranging from the local to the global [22]. By utilizing Java' s object-oriented features, Bayanihan becomes a flexible software framework that makes it

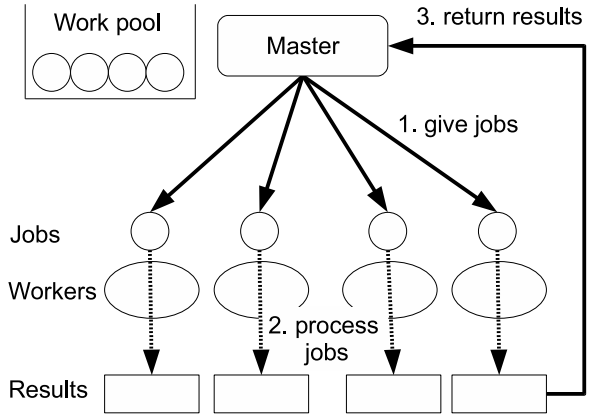


Figure 2. Basic model of VC systems .

easy for programmers to write different VC applications, while allowing researchers to study and develop the underlying mechanisms behind them.

2.3. VC Model

To discuss the performance and reliability of VCs, the behavior of VC systems must be modeled. Here, we provide three models, i.e., basic model for defining computational behavior of VCs, sabotage model and Join/Departure model for workers' behavior. While this section does not cover all detailed behavior of real VC systems, we believe it is nevertheless useful for comparison study.

2.3.1. Basic Model

The basic model of VC is the well known work-pool-based master-worker model [24, 29, 23, 20]. This model is used in almost all VC systems practically. Figure 2 illustrates the model. Details of the model are described as follows.

- A VC system consists of a management node (master) and W different volunteer nodes (workers).
- A computation to be executed in the VC system is divided into N independent jobs.
- At the start of the computation, all jobs are placed in a work pool of the master.
- The master gives a job to each idle worker.
- Each worker executes an allocated job and returns the result to the master. During their execution, no communication exists among workers because jobs are mutually independent.

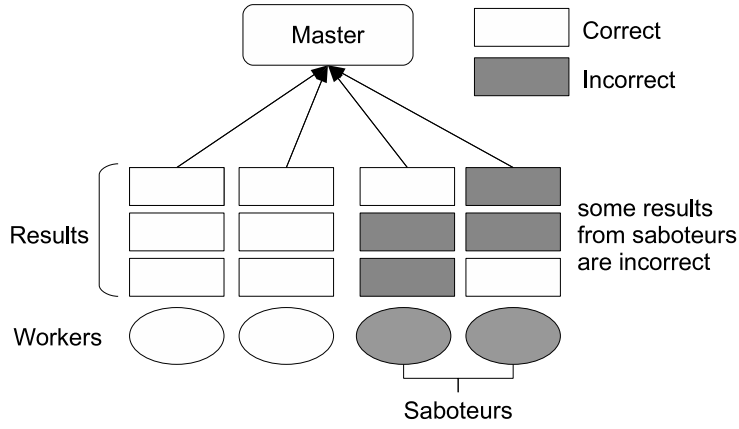


Figure 3. Sabotage model of VC systems .

2.3.2. Sabotage Model

Recently, the need of the sabotage-tolerance is pointed out in [24, 25, 26, 27, 32] because VC allows anyone on the Internet to join a computation. Different from grid computing system which shares managed reliable computers within or among organizations, workers of VC system are owned and managed by general users. The workers may behave erratically due to hardware/software failures or virus infection, or may behave maliciously to falsify the results of jobs, each of which results in sabotage to the computation. It is reported in [27] that a large fraction of workers (about 35 percent) actually returned at least one incorrect results in a real VC.

In addition, the presence of malicious workers (saboteurs) is a serious issue for VC systems to guarantee the computational correctness. Malicious workers can sabotage job execution intentionally by returning erroneous results. Traditional fault-tolerance techniques such as using parity and checksum schemes will not be effective, because they cannot protect against intentional attacks by saboteurs who can disassemble the code. Thus, sabotage-tolerance is a mandatory issue for VC systems in order to make them reliable and viable.

To discuss the sabotage-tolerance problem of VC systems, Sarmenta [23] proposed the following sabotage model of VCs. Figure 3 illustrates the sabotage model.

- A certain faulty fraction f of the W workers are assumed to be saboteurs who might return erroneous (incorrect) results.
- Each saboteur is assumed to return an erroneous result with a constant probability s, which is known as the sabotage rate.
- The values of f and s are unknown to the master.

In VCs with saboteurs, jobs that finish with erroneous results are called erroneous jobs. At the end of the computation, an error rate ϵ can be calculated as the ratio of erroneous jobs to all jobs. The purpose of sabotage-tolerance in VCs is decreasing error rate ϵ as needed for the computation.

Using no sabotage-tolerance mechanisms, the error rate increases in proportion to the number of saboteurs. Let T be the time taken to finish all jobs of a computation, i.e. the

computation time. If all workers function at the same speed and execute a job in a unit time, then T is given by $\lceil N/W \rceil$ unit times and error rate ϵ is given as $N \times f \times s/N = f \times s$. It might be readily apparent that ϵ is proportional to the number of saboteurs and sabotage rate s.

The sabotage model are classified into the following two major classes depending on the values of incorrect results from saboteurs.

- Random attack model [23]
  Random attack model is the simplest sabotage model. In this mode, each saboteur returns a random incorrect result with probability s. The values of those incorrect results are determined at random and never match each other.

- Collusion attack model
  A potential threat of collusion attack comes up in recent VC systems as Zhao et al. [26] acknowledge. In fact, actual developments such as collaborative data distribution [30] or P2P systems [31] are enabling connection of workers. These solutions violate the workers independence assumption, as they enable workers to communicate and collude, thus empowering sabotage.

  In this chapter, we define a collusion attack model as an extension of the random attack model to treat colluding of saboteurs. The saboteurs in the collusion attack model can communicate with each other, and return incorrect results which have the same value to increase the probability that the master accepts their incorrect results. The collusion attack model are classified into the following three categories depending on the colluding method, e.g. how to determine the values of incorrect results.

  - Predefined colluding method
    Predefined colluding is the simplest colluding method. In this method, the values of incorrect results are predefined for each job. Hence, when saboteurs collude, all incorrect results of job j_x have the same incorrect value x. In predefined colluding method, a saboteur colludes with probability c, i.e. colluding rate for each job allocation.

    The values of incorrect results in predefined colluding method are summarized as follows. When job j_x is allocated to a saboteur, the value of incorrect result is set to specified value x with probability $s \times c$ and set to a random value with probability $s \times (1 - c)$.

  - Direct colluding method
    Direct colluding method allows a saboteur to communicate with other saboteur directly and determine the incorrect value for every sabotaging. Thus, the values of incorrect results are dynamically determined by saboteurs who communicate with each other (colluding group). In this method, the number of saboteurs in each colluding group tends to be small since all saboteurs in a colluding group must know informations of all saboteurs in the gruop for direct communications.

  - Indirect colluding method
    Indirect colluding method extends direct colluding to enable building an extensive colluding group of saboteurs. Instead of the direct communication among

saboteurs in a colluding group, the colluded saboteurs access to a third party, e.g. a colluding server, and obtain the value which should be set to the incorrect result. Thus, saboteurs can communicate with each other indirectly (via the colluding server) without knowing informations of other saboteurs. This method has a possibility of allowing all saboteurs to collude together, and increasing error rate of a computation dramatically.

2.3.3. Defection Model

While sabotage-tolerance is a mandatory issue on the reliability of computation, VC also has another issue; that is, the stability of computation [35]. The basic model assumes that workers execute allocated jobs and return their results without fails; however, workers in real VC are never coerced into sustained execution of jobs since they participate in VC systems and provide their resources as volunteers. Even if the allocated job is in execution and not finished, workers can defect from the VC system for any time by terminating the execution and discarding the job. During their execution, workers may have problem due to hardware/software failures (e.g. thermorunaway) or network disconnection, each of which results in defection from the VC system. In addition, after the problem is solved, those workers may rejoin to the VC system. Therefore, for modeling more realistic VCs than the basic model, it must consider those dynamic activities of workers,

Actually, workers in real VC frequently defect and rejoin to the system, as shown in the real traces of VC projects [33, 34]. However, no adequate mathematical-model is provided since the analysis of activities is an extremely-complex problem. In this chapter, we provide a simple mathematical-model, referred to as "random defection model", for discussion about the performance of VC systems with workers' defection. The details of this model are described as follows.

- Two states are defined for each worker, "up" and "down" states. The "up" state represents the worker is active and is able to work for the VC project, whereas the "down" represents the worker is inactive and is not able to work.
  - W_n gets "up" state at T_i with probability $P_{up}(T_i)$.
  - W_n gets "down" state at T_i with probability $P_{down}(T_i)$.
- In each time step, the state of a worker W_n changes as shown in Figure 4.
  - If the state is "up", it changes from "up" to 'down" with probability $utod_n$.
  - If the state is "down", it changes from "down" to "up" with probability $dtou_n$.

$P_{up}(T_i)$ and $P_{down}(T_i)$ are given by following recurrence formulas.

$$P_{up}(T_i) = (1 - utod_n) \times P_{up}(T_{i-1}) + dtou_n \times P_{down}(T_{i-1}) \quad (1)$$

$$P_{down}(T_i) = (1 - dtou_n) \times P_{down}(T_{i-1}) + utod_n \times P_{up}(T_{i-1}) \quad (2)$$

Because W_n gets either up or down state, the sum of $P_{up}(T_i)$ and $P_{down}(T_i)$ is 1 in every time step. From this and the recurrence formulas, the general expressions of $P_{up}(T_i)$ and

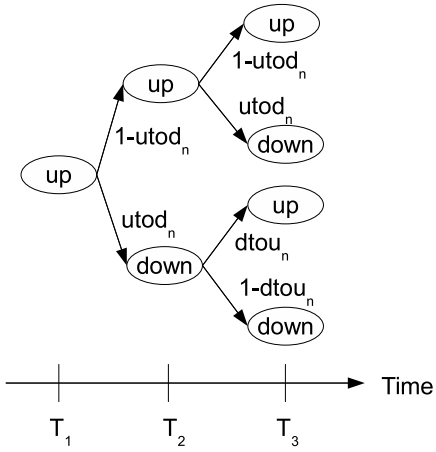


Figure 4. State transition model of worker W_n .

$P_{down}(T_i)$ are given by

$$\begin{aligned} P_{up}(T_i) &= (1 - utod_n - dtou_n)^{i-1} \times (P_{up}(T_1) - \frac{utod_n}{utod_n + dtou_n}) \\ &\quad + \frac{utod_n}{utod_n + dtou_n} \end{aligned} \tag{3}$$

$$\begin{aligned} P_{down}(T_i) &= (1 - utod_n - dtou_n)^{i-1} \times (P_{down}(T_1) - \frac{dtou_n}{utod_n + dtou_n}) \\ &\quad + \frac{dtou_n}{utod_n + dtou_n} \end{aligned} \tag{4}$$

, where $P_{up}(T_1)$ and $P_{down}(T_1)$ represent the initial state of W_n. In the steady-state of W_n (i.e. time step $i \to \infty$), the probability of getting "up" state is given by following expression. Note that $utod_n + dtou_n \neq 0$.

$$P_{up}(T_{\inf}) = \frac{utod_n}{utod_n + dtou_n} \tag{5}$$

Assume that all workers defect and rejoin to the system similarly ($utod_n = p_d$ and $dtou_n = p_u$ for $n = 1, 2, \ldots, W$, where probability p_d is referred to as the defection rate). Since workers' activities are independent of the state of VC projects, we assume that the initial state of the computation is the same as the steady-state of workers. That is, at the start of the computation, the number of workers in the state "up" is $W \times P_{up}(steady)$, where

$$P_{up}(steady) = \frac{p_d}{p_u + p_d}. \tag{6}$$

2.4. Problems and Motivations

As described in this section, many VC projects have run successfully or is currently running on the Internet. Those projects are developed using a VC framework, which provides templates and simplifies the work of creating a VC project. Nowadays it is not so difficult for any scientist to use VC for high performance computing.

While VC becomes easy to use, it still has a mandatory issue for reliable computing; that is, saboteurs in VC systems may behave maliciously and return incorrect results, each of which degrades computational correctness. Against these sabotaging, current VC frameworks provide some sabotage-tolerance mechanisms such as credibility-based voting to decrease error rate of a computation and improve the computational reliability of VC systems. However, those mechanisms are proposed and evaluated in unrealistic VC models such as the simple random attack model without workers' defection. The performance and drawbacks of those mechanisms are hidden in real VC systems. Therefore, current VC frameworks such as BOINC provides the simplest sabotage-tolerance mechanism based on a redundant computation, which significantly degrades the performance instead of ensuring the reliability of computation.

This motivates us to reveal the sabotage-tolerance performance and drawbacks of current sabotage-tolerance mechanisms, and improve them for higher performance and more reliable VC systems. In this chapter, we discuss sabotage-tolerance mechanisms and job scheduling methods used in those toward efficient sabotage-tolerance in real VC systems. First, we evaluate the performance and effectiveness of current sabotage-tolerance mechanisms in more real VC model, e.g. collusion attack model. Then we put the efficiency in perspective of job scheduling problems and propose an sabotage-tolerant job scheduling method which works well in real VC systems.

3. Sabotage-Tolerance Mechanisms and Performance Evaluations

In this section, we will compare some basic mechanisms for sabotage tolerant VC systems. We will first introduce sabotage-tolerance mechanisms such as voting and spot-checking (Sec. 3.1), then compare the performance of those methods in terms of error rate and computation time (Sec. 3.2). We will provide comprehensive comparison study of voting methods, a checking method, and their hybrid methods to reveal their sabotage tolerance performance and drawbacks. We will conclude this section with the brief summary obtained by the comparison study.

3.1. Sabotage-Tolerance Mechanisms

3.1.1. Simple Voting

The basic approach to the sabotage-tolerance of VC systems is voting. Each job is replicated and redundantly allocated to several workers so that a master can collect several results and compare their values. The results collected for a job are then classified into groups (called result groups) according to their values. The master decides which result group should be accepted as the final result of the job through voting. Two major voting methods are M-majority and M-first voting [36].

- M-Majority voting: This method collects M candidate results and accepts the result group which collects the largest number of results as the final result.

- M-first voting: This method collects at least M candidate results and accepts the result group which first collects M matching (the same) results as the final result.

The benefit of these voting methods is its simple principle, which eases its implementation. The M-first voting is supported by the main VC middle-ware BOINC [17, 20] and is widely used by major VC projects [11, 12, 15]. On the contrary, a major weakness lies in inefficient use of huge number of available workers, since the degree of redundancy is static and must be specified by a project owner before starting the computation. The M-majority voting always generates a fixed number of candidate results for each job and the M-first voting always collects a fixed number of matching results, no matter how reliable each result is. The reliability of each worker may differ from one another and change dynamically with time; hence, any statically chosen redundancy value will result in waste of a number of useful workers for unnecessary redundant computations.

If all workers function at the same speed and execute a job in a unit time, the theoretical expectation of error rate of M-first voting is given by eq.(7) [1].

$$\epsilon_{voting}(M, \phi) = \sum_{j=M}^{2M-1} {}_{2M-1}C_j \phi^j (1-\phi)^{2M-1-j}, \tag{7}$$

where ϕ represents the ratio of incorrect results to all produced results. ϕ is equal to $f \times s$ when c, i.e. colluding rate, is equal to 1.

3.1.2. Spot-Checking

Spot-checking [23] is devised to check whether a worker is a saboteur or not directly and is used in web-based VC framework "Bayanihan" [22, 21]. In this technique, a master sometimes assigns a spotter job whose correct result is already known to the master. If a worker returns a result which does not match the correct one, the master can catch the worker as a saboteur.

When the master catches a saboteur by spot-checking, the following two methods can be used:

- Backtracking: The master backtracks (invalidates) all results returned by the saboteur because each might be an incorrect one.

- Blacklisting: The master puts saboteur's identification information (e.g. IP address) on a blacklist to prevent the saboteur from returning results or getting any more jobs.

Backtracking and blacklisting can be used simultaneously for efficient sabotage-tolerance. However, blacklisting is not always effective for VC systems. Although saboteurs can be blacklisted based on their IP addresses, it is not difficult for a saboteur to rejoin the computation as a new worker. Requiring more reliable identification information such as home address will make the blacklisting more effective. However, it also diminishes volunteers' incentive to join the VC system. Based on this trade-off, we must consider both VC systems with and without blacklisting to meet the variant condition of real VCs.

Spot-checking is an effective technique in that master can detect saboteurs with a high degree of confidence unless returned results match the correct one by accident. However,

spot-checking (and similar techniques such as quizzes [26] and sampling [28]) also has some difficulties which must be considered in practical use; master must generate spotter jobs which are not easily identifiable as spotter ones. For this problem, one possible way is implied in [1] that spotter job can be generated by executing a normal job at a reliable node (e.g. master) before starting a computation project. In addition, even if backtracking and blacklisting are used with the spot-checking, using only this technique for sabotage-tolerance purpose will need a huge number of spotter jobs to obtain highly reliable results. An experiment of VC [27] concluded that this technique will require longer computation time than the simple voting like M-first voting for achieving lower error rate than 2×10^{-4} because spot-checking will not efficiently detect saboteurs who return incorrect results infrequently (i.e. saboteurs of small s).

3.1.3. Credibility-Based Voting

Combining the idea of voting and spot-checking to cover their disadvantages, Sarmenta [23] proposes a new voting method, referred to as "credibility-based voting" in this chapter. In this method, each system element such as a worker, result, result group, and job is assigned a credibility value that represents its correctness. Spot-checking is used to estimate worker's credibility. Every worker and result has different credibility, which collectively affect the credibility of the result groups. A job is finished when the credibility of any result group reaches a threshold θ. Therefore, different from M-first voting, the necessary number of results to finish a job is not fixed but rather flexible in accordance with the worker's behaviour. This mechanism reduces unnecessary redundant computations.

Furthermore, this method can guarantee that the mathematical expectation of the error rate is below an arbitrary acceptable error rate ϵ_{acc} for any parameters (i.e. reliability condition $\epsilon \leq \epsilon_{acc}$ is satisfied for any s and f) by setting two conditions: (1) threshold $\theta = 1 - \epsilon_{acc}$, and (2) unknown parameter f satisfies $f \leq f_{max}$, where f_{max} is the upper limit of f and is a known value. This condition implies that the number of saboteurs in W workers is, at most, $f_{max} \times W$.

The details of credibility-based voting is described here with the definitions of the credibility [23]. The credibility of a worker w (denoted by $C_W(w)$) is determined by how many times w survives spot-checking (how many times w returns the correct result for spotter jobs). When blacklisting is used, if worker w_{wb} survives spot-checking k times, $C_W(w_{wb})$ is given by eq. (8).

$$C_W(w_{wb}) = \begin{cases} 1 - f_{max}, & \text{if } k = 0 \\ 1 - \frac{f_{max}}{(1-f_{max}) \times ke}, & \text{otherwise.} \end{cases} \tag{8}$$

Therein, e is Napier's constant. When blacklisting is not used, the credibility of worker w_{wob}, $C_W(w_{wob})$, is given by eq.(9).

$$C_W(w_{wob}) = \begin{cases} 1 - f_{max}, & \text{if } k = 0 \\ 1 - \frac{f_{max}}{k}, & \text{otherwise.} \end{cases} \tag{9}$$

The credibility of a result r produced by worker w (denoted by $C_R(r)$) is equal to $C_W(w)$.

$$C_R(r) = C_W(w). \tag{10}$$

The results collected for a job are grouped into several result groups, $G_1, \ldots, G_g$, each of which includes all results having the same value. The credibility of a result group G_a (denoted by $C_G(G_a)$) is given by eq.(11).

$$C_G(G_a) = \frac{P_T(G_a) \prod_{i \neq a} P_F(G_i)}{\prod_{i=1}^{g} P_F(G_i) + \sum_{n=1}^{g} P_T(G_n) \prod_{i \neq n} P_F(G_i)}, \tag{11}$$

where $P_T(G_a)$ and $P_F(G_a)$ are given as follows;

$$P_T(G_a) = \prod_{r \in G_a} C_R(r), \tag{12}$$

$$P_F(G_a) = \prod_{r \in G_a} (1 - C_R(r)). \tag{13}$$

Therein, $P_T(G_a)$ ($P_F(G_a)$) is the probability that all results in G_a are correct (incorrect). In eq.(11), $C_G(G_a)$ represents the conditional probability that the results in G_a are correct and those in all other groups are incorrect, for a given combination of the result groups.

The credibility of a job j (denoted by $C_J(j)$) is equal to $C_G(G_x)$, where G_x is a result group that has a maximum credibility among all result groups for job j.

$$C_J(j) = C_G(G_x) = \max_{1 \leq a \leq g} C_G(G_a). \tag{14}$$

When $C_J(j)$ reaches threshold θ $(= 1 - \epsilon_{acc})$, the result of the group G_x is accepted as the final result of job j; then job j is finished.

3.2. Performance Evaluation

3.2.1. Simulation Conditions

In this section, we evaluate sabotage-tolerance performance and study the drawbacks of current sabotage-tolerance methods such as M-first voting through the simulation of VCs. Computation times T and error rates ϵ of VCs are evaluated as the average of 100 simulation results. The random (denoted by "random") and the round-robin (denoted by "rr1") methods are used as basic job scheduling methods. The random method selects a job at random and the round-robin method selects a job in a static order, e.g. the order of a job's ID assigned by the master.

The parameters used in our simulation are shown in Table 1. Because some parameters are unknown to the master and are uncontrollable, such as f, s and c, we use variant values for such parameters to simulate various cases of VCs. The upper limit of f, i.e. f_{max}, is set to 0.35 reflecting the result of an experiment in a real VC environment [27]. Because the optimal value of q depends on f and s [38, 39], q is set to 0.1 or 0.2 as in [23, 40].

The simulator developed for this experiment follows that in [23, 40]. At the start of each iteration of simulation, we create a list of W worker entries and randomly select $f \times W$ workers to be saboteurs. We then simulate a computation done by these workers by going through the list in round-robin manner. In each unit time (called "turn"), a worker contacts the master to return a result (for the job it received in the previous turn) and get a new job. This assumes that all workers run at the same speed. To support this approach and generate a stream of pseudo-random numbers, we employ the method Math.random() in Java library, which uses a 48-bit seed and a linear congruential formula [37]. We also assume that a

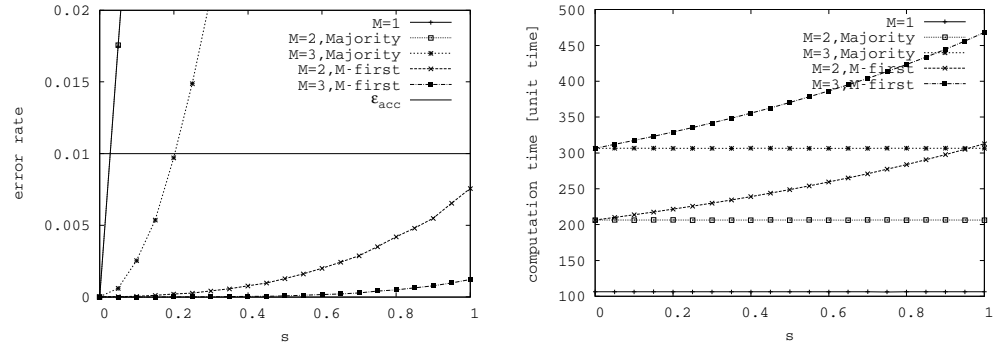
Table 1. Simulation parameters

# of jobs (N)	10000
# of workers (W)	100
acceptable error rate (ϵ_{acc})	0.01 / 0.001
redundancy (M)	$1 \sim 10$
sabotage model	predefined colluding
faulty fraction (f)	$0 \sim f_{max}$ (0.35)
sabotage rate (s)	$0 \sim 1$
colluding rate (c)	$0 \sim 1$
defection model	random defection
defection rate (p_d)	$0 \sim 0.8$
spot-check rate (q)	0.1 / 0.2

saboteur knows when it is caught by spot-checking in a system without blacklisting [23, 40]. After having been caught, the saboteur immediately rejoins to the system and returns incorrect results as a new worker.

3.2.2. M-first Voting vs. M-Majority Voting

First, We provide comprehensive comparison study of two basic voting methods, M-first voting and M-majority voting, to reveal their sabotage tolerance performance and drawbacks.

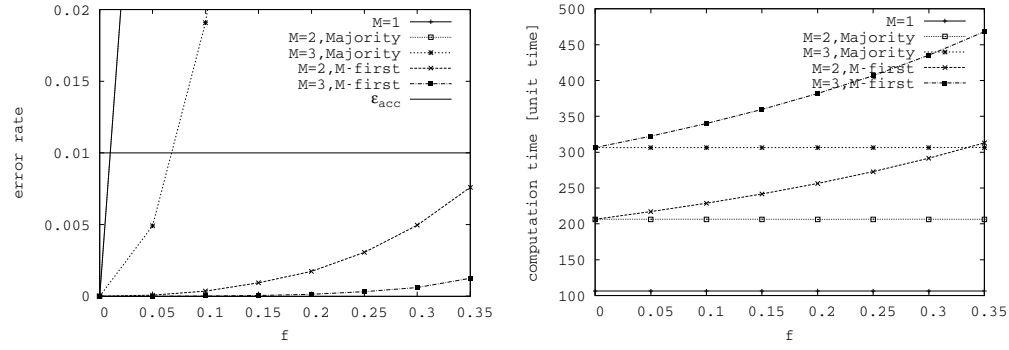


(a) Error-rate ϵ

(b) Computation time T

Figure 5. M-first voting vs. M-majority voting for sabotage rate s ($\epsilon_{acc} = 0.01$, $f = 0.35$, $c = 0.1$, $p_d = 0$, random scheduling).

Sabotage rate s and fraction f Fig.5(a) and Fig.6(a) show error rate of each voting method for sabotage rate s and fraction f, respectively. These figures show that larger



(a) Error-rate ϵ (b) Computation time T

Figure 6. M-first voting vs. M-majority voting for fraction f ($\epsilon_{acc} = 0.01$, $s = 1$, $c = 0.1$, $p_d = 0$, random scheduling).

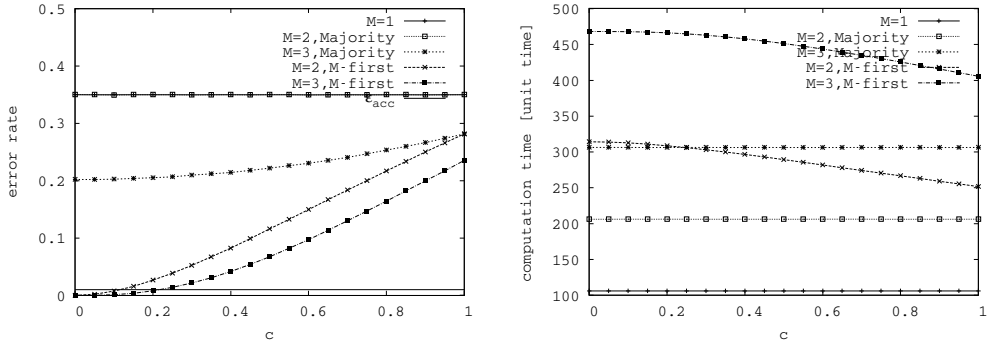
redundancy M decreases the error rate. When $M = 1$, the error rate corresponds to the ratio of the number of incorrect results because all incorrect results are accepted. Thus, the error rate is equal to $f \times s \times N/N = fs$.

In the case of M-majority voting, the error rate for $M = 2$ is equal to that for $M = 1$. This means 2-majority voting does not perform meaningful redundant computation. In 2-majority voting, both of two candidate results are incorrect with probability $(fs)^2$. Also, one candidate result is incorrect and the other is correct with probability ${}_2C_1 \times (fs) \times (1 - fs)$. Since 2-majority voting randomly selects the final result from those candidates in this situation, the incorrect one will be accepted with a probability of 50%. Thus, the error rate of 2-majority voting is equal to $(fs)^2 +_2 C_1 \times (fs) \times (1 - fs) \times 0.5 = fs$. As M becomes larger than three, majority voting performs well based on the principle of majority rule. For example, at $s = 0.2$ and $f = 0.35$ in Fig.5(a), the error rate of 3-majority voting is around 0.01, while $fs = 0.07$.

In the case of M-first voting, the error rate for $M = 2$ is very small compare to that for $M = 1$. This means M-first voting performs good redundant computation and decreases error rate even if the redundancy is the smallest ($M = 2$). Also, for the same redundancy ($M = 3$), the error rate of M-first voting is less than that of M-majority voting for all s. In M-first voting, incorrect results having the different values are difficult to be accepted as the final result because M-first voting collects M matching results even if it already holds M candidate results. This is why the error rate of M-first voting is less than that of M-majority voting for the same M.

Fig.5(b) and Fig.6(b) show computation time of each voting method for sabotage rate s and fraction f, respectively. The computation time of M-majority voting stays constant for all s and f because each job finishes with just two results whether the results are correct or not. Thus, the computation times are around $M \times N/W = 200$ for $M = 2$ and 300 for $M = 3$, respectively. (There are minor differences between the simulation results and those values since some jobs may have more than two results depending on the sequences of job allocations and completion determinations.)

Contrary to the case of M-majority voting, the computation times of M-first voting depends on the value of s and f. At $s = 0$ or $f = 0$, the computation times of M-first voting are the same as those of M-majority voting because each job finishes with M results, each of which has the correct value and matches each other. If there are incorrect results, each job may have more than M results, which increases the computation time. The mean number of collected results increases with the ratio of the number of incorrect results to all results. Thus, as s and f increase, the computation times of M-first voting also increase.



(a) Error-rate ϵ (b) Computation time T

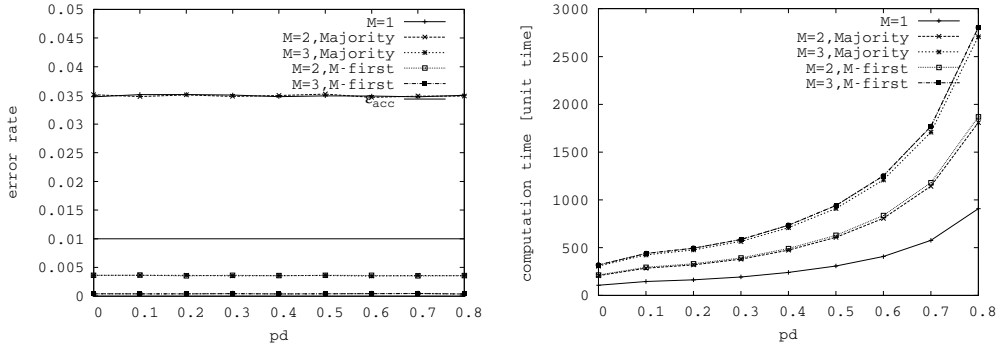
Figure 7. M-first voting vs. M-majority voting for colluding rate c ($\epsilon_{acc} = 0.01$, $f = 0.35$, $s = 1$, $p_d = 0$, random scheduling).

Colluding rate c Fig.7(a) shows error rate of each voting method for colluding rate c. This figure shows error rates of both M-majority and M-first voting methods increase with c. Since the number of matching incorrect results increases in proportion to c, those results tend to be accepted as majority ones. When c is small, the values of incorrect results do not match each other, resulting in small error rate in M-first voting. However, the error rate of M-first voting dramatically increases as c increases. This result indicates that M-first voting works well only when c is small.

Fig.7(a) also shows that error rates of 2-first voting and 3-majority voting are the same at $c = 1$. In 3-majority voting, some one of result groups G_X has more than 2 results because there are only two result groups, i.e. the group of correct results and the group of incorrect (matching) results. Since the number of results in the other group is smaller than 2, G_X is accepted as the final result. Here, 2-first voting also accepts G_X regardless of the order of results production; thus, the error rates of 2-first voting and 3-majority voting are the same. Generally, the error rates of M-first voting and $(2M-1)$-majority voting are the same at $c = 1$, while the computation time of M-first voting is smaller since the number of required results is from M to $(2M-1)$.

Fig.7(b) shows computation time of each voting method for colluding rate c. The computation time of M-majority voting stays constant for any c as the cases of s and f because each job finishes with just M results. On the other hand, in cases of M-first voting, the computation time changes with the value of c. This is because there are some jobs finished

with M incorrect results when saboteurs collude. Since larger c increases the number of such jobs, it decreases the computation time, while increasing error rate.



(a) Error-rate ϵ (b) Computation time T

Figure 8. M-first voting vs. M-majority voting for defection rate p_d ($\epsilon_{acc} = 0.01$, $s = 0.1$, $c = 1$, $P_{up}(steady) = 0.8$, random scheduling).

Defection rate p_d Fig.8 shows error rate and computation time of each voting method for defection rate p_d under the condition of $P_{up}(steady) = 0.8$. The value of p_u is set corresponding to p_d so that 80% of workers are participating in the system, on average. For instance, when $p_d = 0.1$, p_u is set to 0.4. This figure shows error rates of voting methods stay constant, while computation times increase with p_d.

In the random defection model shown in Sec. 2.3.3, the expectation of the computation time for M-majority voting is given as following. By supposing each worker has the same property, the mean number of active state workers W_{up} at each time step is given by eq.(15). Note that $P_{up}(steady)$ is constant.

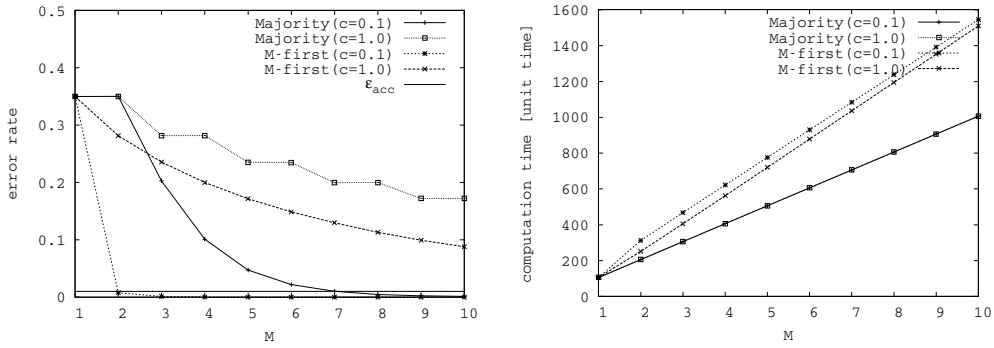
$$\begin{aligned} W_{up} &= W \times P_{up}(steady) \\ &= W \times \frac{p_d}{p_u + p_d}. \end{aligned} \tag{15}$$

When a worker W_n gets a job which takes i unit times to process, W_n returns a result for the job with probability $P(i) = (1 - utod_n)^i$. If all workers function at the same speed and return results within 1 unit time, W_{up} workers get jobs in each unit time and return their results with probability $P(1)$ on average. Since M results are collected for all N jobs, the expectation of the computation $T_{majority}(M)$ is given by following.

$$\begin{aligned} T_{majority}(M) &= M \times \frac{N}{W_{up} \times P(1)} \\ &= M \times \frac{N}{W \times P_{up}(steady)} \times \frac{1}{1 - p_d}. \end{aligned} \tag{16}$$

Eq.(16) shows the computation time is proportional to $1/(1 - p_d)$. The computation time of M-first voting also increases with p_d as shown in Fig.8(b). Thus, the actual values

of computation times in real VCs become larger than those in the basic model (i.e. $p_d = 0$), since workers in real VCs frequently defect and rejoin the system corresponding with non-zero p_d. For example, when $p_d = 0.5$, it doubles the computation times compared with those in the basic model.



(a) Error-rate ϵ

(b) Computation time T

Figure 9. M-first voting vs. M-majority voting for redundancy M ($\epsilon_{acc} = 0.01$, $f = 0.35$, $s = 1$, $p_d = 0$, random scheduling).

Redundancy M Fig.9 shows error rate and computation time of each voting method for redundancy M. This figure shows error rates of both M-majority and M-first voting methods decrease with M, while increasing the computation times.

This figure shows that M-first voting decreases error rate efficiently when c is small (e.g. $c = 0.1$). For example, two correct results (i.e. 2-first voting) is enough to satisfy the condition $\epsilon \leq \epsilon_{acc} = 0.01$ in this case. On the other hand, M-majority voting does not decrease error rate as M-first voting and requires huge redundancy ($M = 7$ in this case) to satisfy the condition $\epsilon \leq \epsilon_{acc}$. This means that the computation time of M-majority voting is huge (e.g. $T = 700$ at $M = 7$), and that of M-first voting is small (e.g. $T = 300$ at $M = 2$) for the same condition $\epsilon_{acc} = 0.01$.

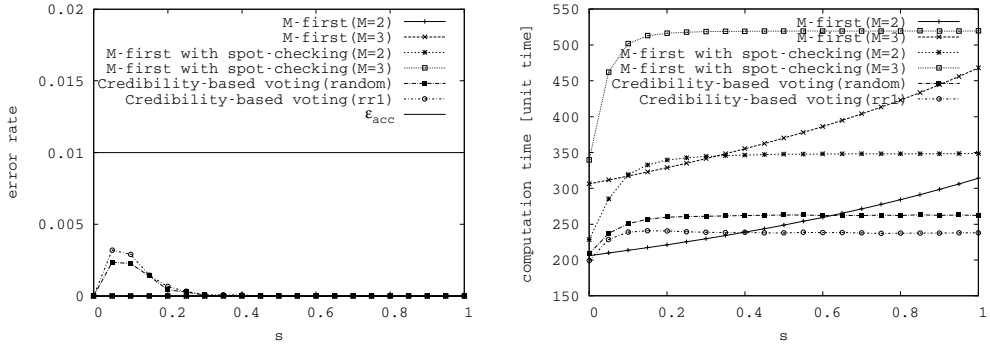
This figure also shows that, when c is large, both M-majority and M-first voting methods do not decrease error rate efficiently and require huge redundancy to satisfy the reliability condition $\epsilon \leq \epsilon_{acc}$. Even if $M = 10$, error rates of both methods are around 0.1 or larger when $c = 1$. The required values of M obtained from eq.(7) for M-first voting are $M = 15$ for $\epsilon_{acc} = 0.05$, $M = 29$ for $\epsilon_{acc} = 0.01$ and over 50 for $\epsilon_{acc} = 0.001$, respectively. This result indicates that both M-majority and M-first voting methods work well only when c is small. Thus, there are needs to introduce such a novel sabotage-tolerance method that efficiently decreases error rates even if saboteurs frequently collude.

3.2.3. M-First Voting with Spot-Checking vs. Credibility-Based Voting

Here, we evaluate the performance of spot-checking based methods. As spot-checking can remove saboteurs from the systems by directly checking their behavior, this technique

seems to efficient to enhance the reliability of computation. Spot-checking can be simply combined with the simple voting methods. We focus on the hybrid method of M-first voting and spot-checking, i.e. M-first voting with spot-checking, and credibility-based voting. The relation among those sabotage-tolerance methods are summarized as following.

- simple voting methods
  The master allocates a job to several workers for majority decision.
  - M-majority voting
  - M-first voting
- spot-checking-based methods
  The master allocates spotter jobs with probability q in addition to allocate normal jobs.
  - M-first voting with spot-checking (M-FVSC in short)
  - Credibility-based voting



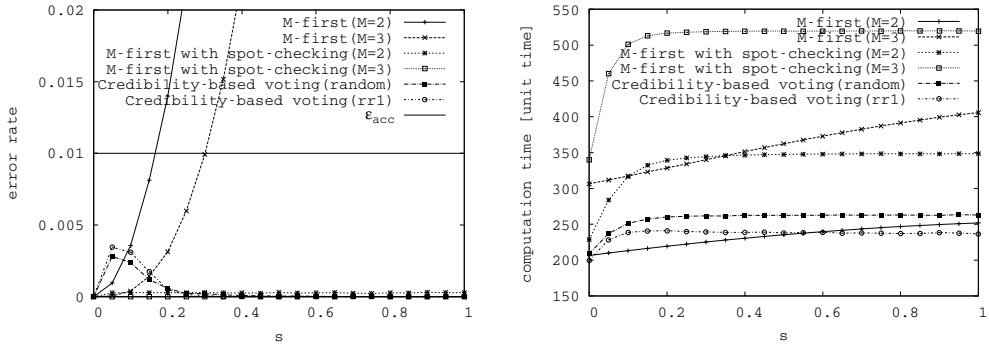
(a) Error-rate ϵ (b) Computation time T

Figure 10. M-first voting with spot-checking vs. Credibility-based voting for sabotage rate s ($\epsilon_{acc} = 0.01$, $f = 0.35$, $c = 0.0$, $q = 0.1$, $p_d = 0$, random scheduling with blacklisting).

Sabotage rate s in cases with blacklisting Fig.10 (a) shows error rate of each method for sabotage rate s when no saboteurs collude ($c = 0$). This figure shows that each sabotage-tolerance method guarantees the reliability condition $\epsilon \leq \epsilon_{acc} = 0.01$ when $c = 0$. The error rates of M-first voting and M-FVSC are 0. This is true because every incorrect result has a different value and no incorrect result will be accepted as majority. On the other hand, in credibility-based voting, some incorrect results can be accepted as final ones, which increases the error rate of the computation. This is because credibility-based voting reduces the number of redundant results as much as possible (as long as the error rate does not exceed ϵ_{acc}) for higher performance of VC systems.

Fig.10 (b) shows computation time of each method. Compared to M-first voting, M-FVSC takes longer computation time for the same M. Due to spot-checking, the computation time increases by following two reasons. (1) Spotter jobs allocated to workers are extra ones, which wastes worker resources. It increases total amount of jobs and computation time by $1/(1-q)$ times. (2) Saboteurs detected by spot-checking and results produced by those saboteurs are eliminated from the system. This reduces the number of workers and the number of results produced in each unit time.

Fig.10 (b) also shows that, in cases with blacklisting, computation times of spot-checking-based methods (credibility-based voting and M-FVSC) stay constant for larger s. Since spot-checking detects a saboteur with probability s, all saboteurs can be detected and eliminated from the system when s is larger than a certain value. All results produced by the saboteurs are invalidated regardless of the value of s. Therefore, sabotage-rate s does not affect on the computation time so much, when blacklisting is used.



(a) Error-rate ϵ

(b) Computation time T

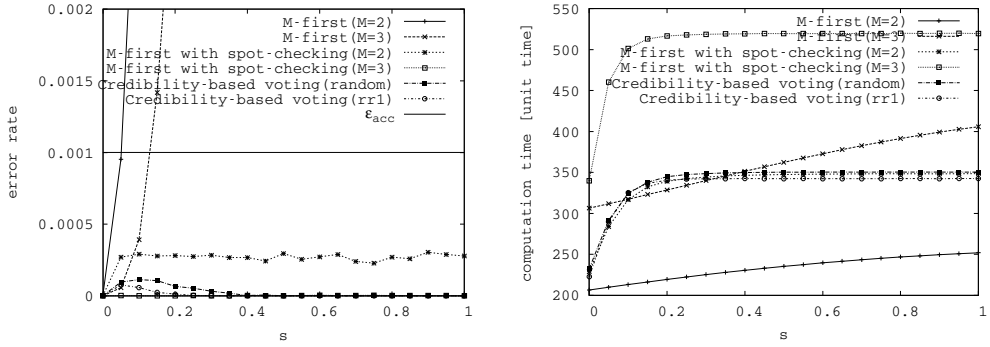
Figure 11. M-first voting with spot-checking vs. Credibility-based voting for sabotage rate s ($\epsilon_{acc} = 0.01$, $f = 0.35$, $c = 1.0$, $q = 0.1$, $p_d = 0$, random scheduling with blacklisting).

Fig.11 (a) shows error rates of each method for sabotage rate s when saboteurs collude ($c = 1$). Different from the case of $c = 0$ (Fig.10 (a)), the error rate of M-first voting becomes quit large. On the other hand, the error rates of spot-checking-based methods are smaller since spot-checking eliminates incorrect results from saboteurs. This result indicates that spot-checking is a promising approach to the reduction of error rates against the collusion attack method.

Fig.10 (a) and Fig.11 (a) show that the error rates of spot-checking-based methods are smaller than ϵ_{acc} for any s, whether saboteurs collude ($c = 1$) or not ($c = 0$). Although the values of s and c are unknown to the master, those method can guarantee the reliability condition $\epsilon \leq \epsilon_{acc} = 0.01$ in this case.

From Fig.10 (b) and Fig.11 (b), it seems that credibility-based voting outperforms M-FVSC. For satisfying the condition $\epsilon \leq \epsilon_{acc} = 0.01$ in cases with blacklisting, there is a large difference of the computation time among those methods. For example, the computation time of credibility-based voting with round-robin scheduling is around 240 at $s = 1$, while that of 2-FVSC is around 350.

The performance difference between credibility-based voting and M-FVSC comes from the difference of the required number of results to finish a job. In M-FVSC, all jobs require at least M results, even if workers survive a number of spot-checking and seem to be non-saboteurs. On the other hand, in credibility-based voting, workers who survive a number of spot-checking have large credibility and produce large credibility results. If the credibility of a result is large enough (larger than threshold $\theta = 1 - \epsilon_{acc}$), a job having the result can be finished with only that result, i.e. the redundancy is 1. Thus, the mean number of the required results to finish jobs can be smaller than two. This is why credibility-based voting shows better performance compared to 2-FVSC.

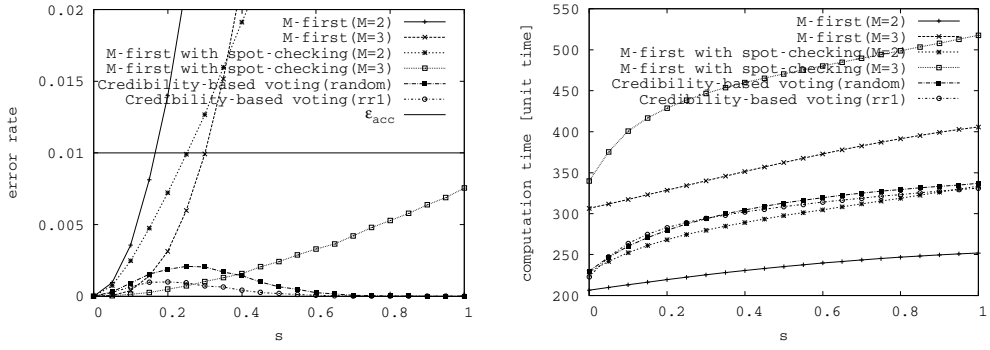


(a) Error-rate ϵ (b) Computation time T

Figure 12. M-first voting with spot-checking vs. Credibility-based voting for sabotage rate s ($\epsilon_{acc} = 0.001$, $f = 0.35$, $c = 1.0$, $q = 0.1$, $p_d = 0$, random scheduling with blacklisting).

Fig.12 shows error rate and computation time of each method at $\epsilon_{acc} = 0.001$. Even if ϵ_{acc} changes from 0.01 to 0.001, spot-checking-based methods can guarantee the reliability condition $\epsilon \leq \epsilon_{acc}$ for any s. Note that the computation time of credibility-based voting depends on ϵ_{acc}. As shown in Fig.11 (b) and Fig.12 (b), the computation time of credibility-based voting becomes larger for smaller ϵ_{acc}. For smaller ϵ_{acc}, the number of jobs finished with only one result becomes smaller, since workers who have enough high credibility (larger than $1 - \epsilon_{acc}$) become smaller. When $\epsilon_{acc} = 0.001$, almost all jobs require two results in credibility-based voting; thus, the computation times of both credibility-based voting and 2-FVSC become almost the same.

Sabotage rate s in cases without blacklisting Fig.13 (a) shows the error rate of each method for sabotage rate s. This figure shows that, even in cases without blacklisting, credibility-based voting guarantees the reliability condition $\epsilon \leq \epsilon_{acc} = 0.01$. This is true because the calculation formula of credibility changes in accordance with the availability of blacklisting. If blacklisting is not available, a saboteur can rejoin to the system and produce incorrect results permanently. Thus, the credibility given to each worker, i.e. the probability of returning correct results, becomes smaller than that in cases with blacklisting as shown in eqs.(8) – (9).



(a) Error-rate ϵ (b) Computation time T

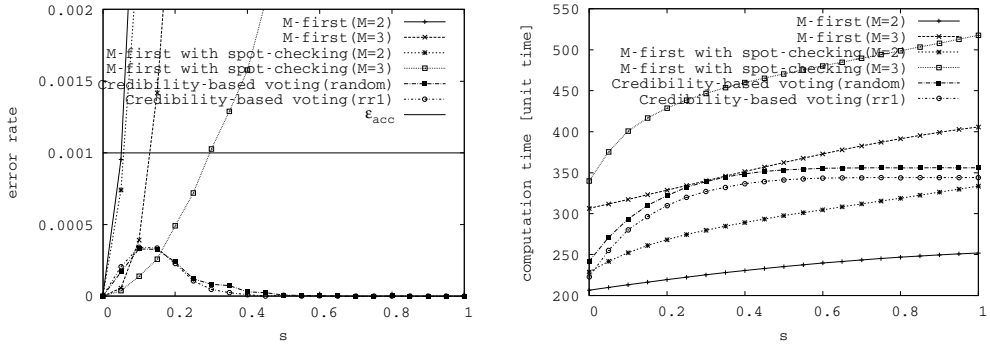
Figure 13. M-first voting with spot-checking vs. Credibility-based voting for sabotage rate s ($\epsilon_{acc} = 0.01$, $f = 0.35$, $c = 1.0$, $q = 0.1$, $p_d = 0$, random scheduling without blacklisting).

Fig.13 (a) also shows that error rates of M-FVSC increase in proportion to s and exceed ϵ_{acc} when $s \geq 0.25$ and $M = 2$. This result indicates that only using spot-checking is inadequate to guarantee the reliability condition $\epsilon \leq \epsilon_{acc}$. It requires redundant computation with larger redundancy ($M \geq 3$ in this case) to satisfy the reliability condition for any s. Although spot-checking detects saboteurs and eliminates incorrect results, in cases without blacklisting, those saboteurs rejoin to the system and produce incorrect results with probability s, which remain in the system until the end of the computation. Since the number of remaining incorrect results is proportional to s, the error rate increases with s and reaches its peak at $s = 1$. Thus, to guarantee the reliability condition for any s, the redundancy M should be set to a larger value ($M = 3$) supposing the worst case $s = 1$.

Fig.13 (b) shows the computation time of each method. Different from the cases with blacklisting (Fig.11), the computation times of spot-checking-based methods increase with s. This is true because results from saboteurs remain in the system as candidate results of voting, even if s is large. For larger s, saboteurs are easily detected and eliminated from the system; however they return to the system and produce their results in cases without blacklisting. The number of remained incorrect results which do not match with correct ones increases with s, while decreasing the number of correct results produced in each unit time. Thus, in cases without blacklisting, larger s decreases the number of correct results which finish jobs, and increases the computation time.

Fig.13 (b) also shows that the computation time of credibility-based voting is almost the same as that of 2-FVSC at $\epsilon_{acc} = 0.01$. In this case, almost all jobs require two results in credibility-based voting like as the case of Fig.12. For $\epsilon_{acc} = 0.01$, M-FVSC requires the redundancy at least $M = 3$ to satisfy the reliability condition for any s. Also, this figure shows that credibility-based voting outperforms 3-FVSC. The differences of computation times between those methods range from 110 ($T = 340$ and $T = 230$ at $s = 0$) to 190 ($T = 520$ and $T = 330$ at $s = 1$).

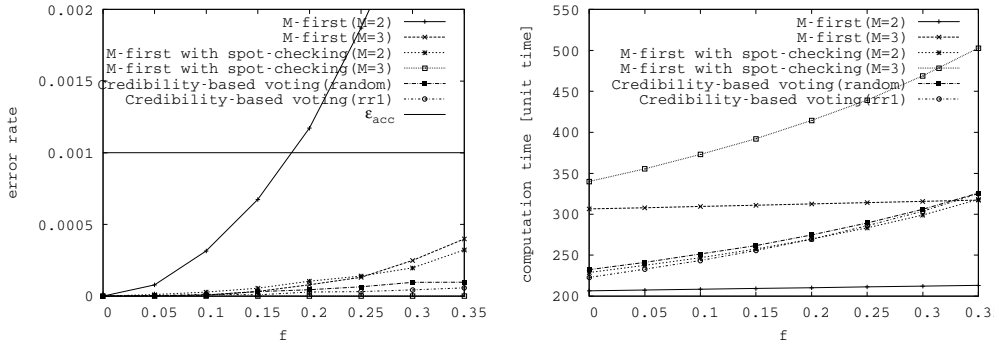
Fig.14 shows the error rate and computation time of each method at $\epsilon_{acc} = 0.001$. This



(a) Error-rate ϵ (b) Computation time T

Figure 14. M-first voting with spot-checking vs. Credibility-based voting for sabotage rate s ($\epsilon_{acc} = 0.001$, $f = 0.35$, $c = 1.0$, $q = 0.1$, $p_d = 0$, random scheduling without blacklisting).

figure shows that, even if $M = 3$, M-FVSC can not guarantee the reliability condition, while credibility-based voting achieves it for any s. It requires larger redundancy ($M \geq 4$) for M-FVSC, which results in increased computation time. Also, the computation time of credibility-based voting is smaller than that of 3-FVSC. This result indicates that credibility-based voting is the most promising approach of spot-checking-based sabotage-tolerance methods for high-performance and reliable VC systems.



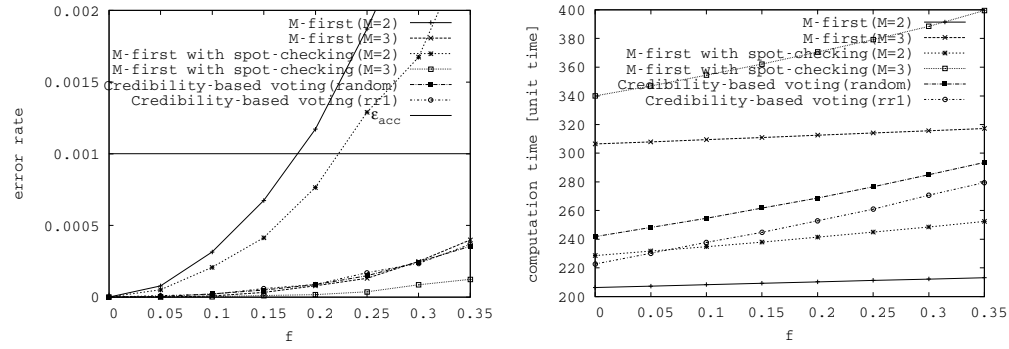
(a) Error-rate ϵ (b) Computation time T

Figure 15. M-first voting with spot-checking vs. Credibility-based voting for fraction f ($\epsilon_{acc} = 0.001$, $s = 0.1$, $c = 1.0$, $q = 0.1$, $p_d = 0$, random scheduling with blacklisting).

Fraction f Fig.15 (a) shows the error rates of each method for fraction f in cases with blacklisting. This figure shows that both spot-checking-based methods guarantee the relia-

bility condition $\epsilon \leq \epsilon_{acc}$ for any f. In contrast to the case of sabotage rate s (Fig.12), error rates of spot-checking-based methods increase with f, i.e. the number of saboteurs. This is true because, as f becomes larger, more saboteurs survive spot-checking and produce incorrect results, which remain in the system until the end of the computation. Since the number of remaining incorrect results is proportional to f, the error rate increases with f and reaches its peak at $f = f_{max}$. Thus, to guarantee the reliability condition for any f, the redundancy M should be set to a larger value supposing the worst case $f = f_{max}$.

Fig.15 (b) shows the computation time of each method. The computation time of M-first voting stays almost constant for any f, while those of spot-checking-based methods increase with f because the results produced by detected saboteurs are invalidated. As f increases, more saboteurs are detected and eliminated from the system, resulting in decrement of the number of valid results. This result indicates that the value of f has a significant impact on the computation time of spot-checking-based methods.



(a) Error-rate ϵ (b) Computation time T

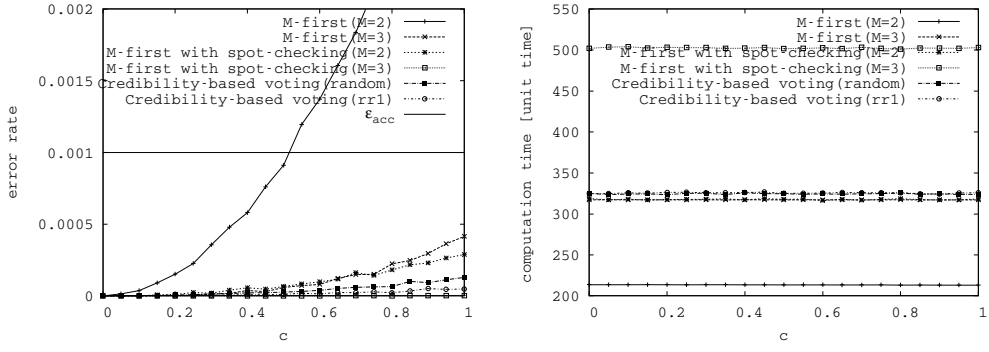
Figure 16. M-first voting with spot-checking vs. Credibility-based voting for fraction f ($\epsilon_{acc} = 0.001$, $s = 0.1$, $c = 1.0$, $q = 0.1$, $p_d = 0$, random scheduling without blacklisting).

Fig.16 shows the error rate and the computation time of each method in cases without blacklisting. This figure shows that error rates of both 2-first voting and 2-FVSC exceed ϵ_{acc} for $f \geq 0.2$. Since the value of f is unknown beforehand, redundancy (i.e. M) should be set to 3 supposing the worst case $f = f_{max}$. Note that larger M increases the computation time, while decreasing error rate. From Fig.16 (b), it is clear that the computation times of 3-first voting and 3-FVSC are larger than that of credibility-based voting. For example, the computation time of 3-first voting is around 310, while that of credibility-based voting using round robin scheduling is around 280. This result indicates that M-FVSC requires huge redundancy M to guarantee the reliability condition for any s and f since it must suppose the worst case ($s = 1$ and $f = f_{max}$).

Fig.16 (b) also shows that, when f is very small, credibility-based voting may be slow. For example, when $f = 0$, the computation time of 2-FVSC is smaller than that of credibility-based voting, while the error rates of both methods are 0 at $f = 0$. This result indicates that credibility-based voting may perform excess redundant computation when f

is very small. This is true because credibility-based voting calculates credibility and determine the redundancy supposing that the values of unknown parameters s and f are the worst values ($f = f_{max}$ etc.) to guarantee the reliability condition for any combination of s and f. Therefore, in some cases where error rate is small, credibility-based voting may perform redundant computation excessively and degrade the performance of VC systems, as M-first voting with larger M does.

One of the solution to ease this performance degradation is focusing on job scheduling methods of credibility-based voting. Since credibility-based voting with any scheduling method guarantees the reliability condition, there is no drawback to use faster job scheduling method (round robin method in this case). As shown in Fig.16 (b) and others, there is a clear difference between the computation time of round robin method and that of random method. Thus, we present a novel job scheduling method which enhances the performance of credibility-based voting in the next section.

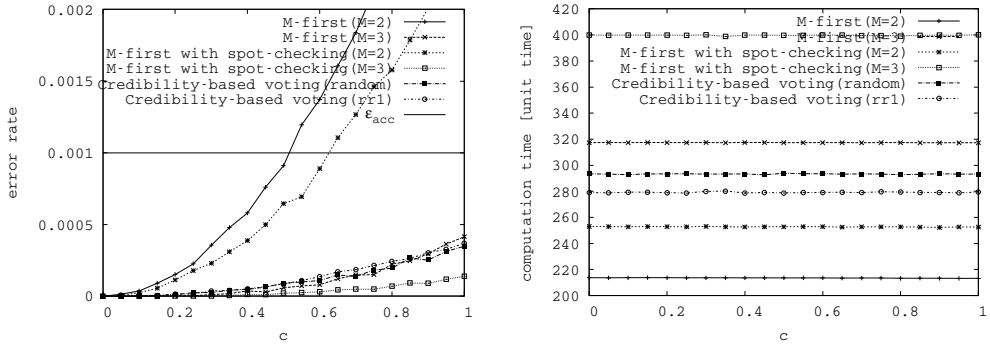


(a) Error-rate ϵ (b) Computation time T

Figure 17. M-first voting with spot-checking vs. Credibility-based voting for colluding rate c ($\epsilon_{acc} = 0.001$, $s = 0.1$, $f = 0.35$, $q = 0.1$, $p_d = 0$, random scheduling with blacklisting).

Colluding rate c Fig.17 shows error rate and computation time of each method for colluding rate c in cases with blacklisting. Error rates of all methods increase with c, because larger c increases the number of matching incorrect results and the probability of their being final result. The value of c is also unknown to the master beforehand like s and f. Thus, to guarantee the reliability condition for any c, the redundancy M should be set to a larger value by supposing the worst case $c = 1$. Note that credibility-based voting guarantees the reliability condition for any c. The computation times of spot-checking-based methods stay constant for any c because almost all saboteurs are eliminated from the system regardless the value of c.

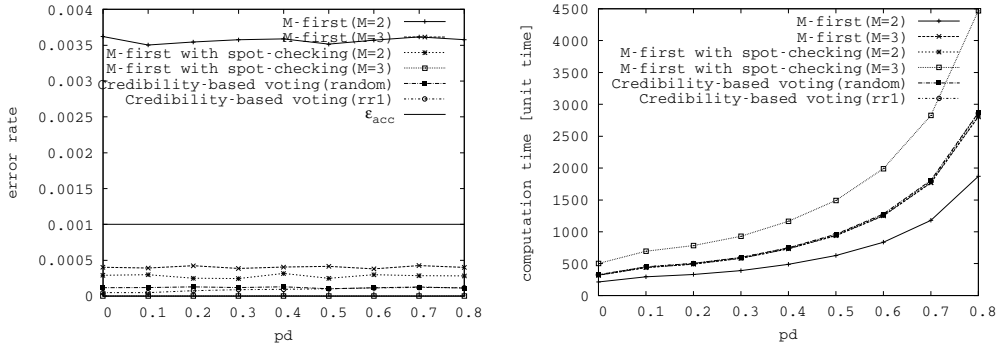
Fig.18 shows error rate and computation time of each method in cases without blacklisting. This figure shows that the error rates of both 2-first voting and 2-FVSC exceed ϵ_{acc} when $c \geq 0.65$. The redundancy M should be set to larger than 3; however, such large



(a) Error-rate ϵ (b) Computation time T

Figure 18. M-first voting with spot-checking vs. Credibility-based voting for colluding rate c ($\epsilon_{acc} = 0.001$, $s = 0.1$, $f = 0.35$, $q = 0.1$, $p_d = 0$, random scheduling without blacklisting).

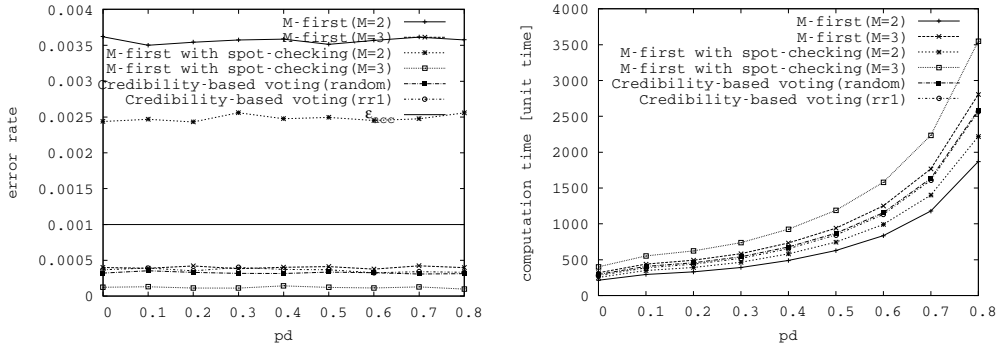
M degrades the performance of VC systems. For example, the computation time of 3-first voting is around 320, while that of credibility-based voting ranges from 280 to 295.



(a) Error-rate ϵ (b) Computation time T

Figure 19. M-first voting with spot-checking vs. Credibility-based voting for defection rate p_d ($\epsilon_{acc} = 0.001$, $s = 0.1$, $f = 0.35$, $q = 0.1$, $c = 1.0$, $P_{up}(steady) = 0.8$, random scheduling with blacklisting)

Defection rate p_d Fig.19 and Fig.20 show error rate and computation time of each method for defection rate p_d at $P_{up}(steady) = 0.8$. This figure shows error rates of all methods stay constant, while computation times increase with p_d. As shown in eq.(16), the computation time is proportional to $1/(1-p_d)$. Thus, the actual values of computation times in real VCs become larger than those in the basic model ($p_d = 0$), since workers in real VCs frequently



(a) Error-rate ϵ

(b) Computation time T

Figure 20. M-first voting with spot-checking vs. Credibility-based voting for defection rate p_d ($\epsilon_{acc} = 0.001$, $s = 0.1$, $f = 0.35$, $q = 0.1$, $c = 1.0$, $P_{up}(steady) = 0.8$, random scheduling without blacklisting).

defect and rejoin the system, which resulting in non-zero p_d. For example, when $p_d = 0.5$, it doubles the computation times compared with those in the basic model.

3.3. Conclusion

In this section, we compared the performances of M-majority voting, M-first voting, M-FVSC and credibility-based voting in terms of error rate and computation time of VCs. Through the extensive simulations, we found the following important results;

- Error rate:
  - M-first voting decreases error rate more efficiently than M-majority voting.
  - When saboteurs collude, error rate dramatically increases in simple voting methods such as M-first voting. Spot-checking-based methods, i.e. M-FVSC and credibility-based voting, work well in this case.
  - When s, f and c are large, M-FVSC requires huge redundancy to satisfy the reliability condition $\epsilon \leq \epsilon_{acc}$.
  - Credibility-based voting guarantees the reliability condition for any s, f, c and p_d.
- Computation time:
  - Credibility-based voting tends to show better performance than M-FVSC; however, it sometimes performs excess redundant computation and degrades the performance since it always assumes the worst case (e.g. $f = f_{max}$ and $c = 1$).
  - The computation time of credibility-based voting largely depends on the job scheduling method.

– The actual values of computation times in real VCs become larger than those in the basic model (i.e. $p_d = 0$).

Thus, as shown clearly in the comparison study, devising an efficient job scheduling method for credibility-based voting is one feasible approach for realizing high-performance and reliable VC systems.

4. Expected-Credibility-Based Job Scheduling Method

As shown in the comparison study of the previous section, credibility-based voting is the most promising approach for high-performance and reliable VC systems since it guarantees the reliability condition for any situation and tends to show better performance than M-FVSC. Also, the computation time of credibility-based voting largely depends on the job scheduling method. The original scheduling method used in [23], i.e. round robin method, is not efficient. Therefore, devising an efficient scheduling method for credibility-based voting is one feasible approach for realizing high-performance and reliable VC systems.

In this section, we propose a dynamic job scheduling method for credibility-based voting, which is referred as "expected-credibility-based job scheduling" in this chapter. We will first discuss the job scheduling problem in credibility-based voting (Sec. 4.1), then propose expected-credibility-based job scheduling method (Sec. 4.2). Then, we will provide comparison study of proposed and other job scheduling methods to reveal their performance (Sec. 4.3).

4.1. Job Scheduling Problems and Motivations

A sabotage-tolerance mechanism generally requires some extra computation time (overhead) because of a redundant computation. For example, when each job is replicated and allocated to M workers for a voting, it increases the overall computation time by M times compared to the case of non-redundant computation. Similarly, spot-checking with the spot-check rate q [23] increases the computation time by $1/(1-q)$ times.

In basic voting methods, the necessary number of results for each job is fixed in advance (e.g. M matching results are required in M-first voting). In this case, the order of execution of jobs does not affect the total number of results produced for a computation. Even if the order of execution of jobs is changed, the overall computation time remains unchanged.

In contrast, with credibility-based voting, the necessary number of results for each job is not fixed in advance because it depends on the credibility of each result. The credibility of result changes as the computation proceeds, depending on the result of spot-checking. In this case, the order of execution of jobs directly affects the total number of results.Therefore, in credibility-based voting, job scheduling method have a considerable impact on the overhead and the computation time of the computation as shown in the previous section.

In the sabotage-tolerant VC systems using credibility-based voting, the job scheduling problem is summarized as follows:

> *select a job that should be allocated to an idle worker prior to other jobs for reducing computation time T to the greatest extent possible, while guaranteeing the computational correctness as $\epsilon \leq \epsilon_{acc}$ for any given ϵ_{acc}.*

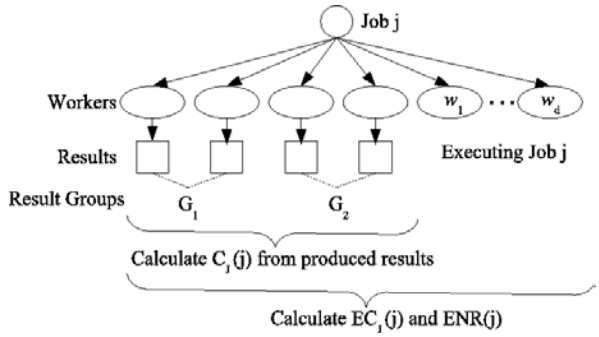


Figure 21. Calculation of $EC_J(j)$ and $ENR(j)$.

Basic job scheduling methods for credibility-based voting are the random and round-robin methods [23]. The random method selects a job at random and the round-robin method selects a job in a static order, e.g. the order of a job's ID assigned by the master. Although these job scheduling methods are simple, they are not efficient because they do not take account of a job's progress.

Thus, we propose a new job scheduling method for credibility-based voting that can consider the progress of each job to reduce the overhead and computation time T. We define two novel metrics for each job: the expected credibility and the expected number of results. Using these two metrics, the proposed scheduling method can select a proper job to be executed prior to others, thereby achieving a reduction in the computation time of a computation. The proposed method employs spot-checking with a constant rate q and backtracking, as in credibility-based voting [23].

4.2. Expected-Credibility-Based Job Scheduling

4.2.1. Definitions of Expected Credibility

To select a job for prior execution, it is important to consider the progress of each job. The simple metric to represent the progress might be the credibility of the job because a job is finished when the credibility of the job reaches threshold θ. However, this metric is insufficient to grasp the proper progress because there might be several workers who are engaged simultaneously in the execution of the job. Consider the case in which job j is allocated to several workers, as shown in Fig.21. According to eq.(14), $C_J(j)$ is calculated from the credibility of returned results. In the calculation, the workers who are currently executing job j are not examined ($w_1, \ldots, w_d$ in Fig.21). Those workers will return their results, which affect $C_J(j)$; therefore, the consideration of such workers is necessary to grasp the proper progress of jobs.

Based on the idea described above, we define two new metrics for each job: the expected number of results and the expected credibility. Presuming that job j has several results which can be grouped into $G_1, \ldots, G_g$ and presuming that there exist d workers ($w_1, \ldots, w_d$) who are executing job j as shown in Fig.21, then the expected number of results and the expected credibility of job j, denoted by $ENR(j)$ and $EC_J(j)$ respectively, are defined as follows.

- The expected number of results (ENR)
  $ENR(j)$ is defined as

$$ENR(j) = \sum_{i=1}^{g} |G_i| + d, \tag{17}$$

  where $|G_i|$ represents the number of results in G_i.

  The metric $ENR(j)$ represents the number of results of job j when all d workers return their results for job j.

- Expected credibility (EC_J)
  $EC_J(j)$ is defined as

$$EC_J(j) = \max_{1 \le a \le g} C'_G(G_a), \tag{18}$$

  where

$$C'_G(G_a) = \frac{P'_T(G_a) \prod_{i \neq a} P'_F(G_i)}{\prod_{i=1}^{g} P'_F(G_i) + \sum_{n=1}^{g} P'_T(G_n) \prod_{i \neq n} P'_F(G_i)},$$

$$P'_T(G_a) = \begin{cases} P_T(G_a), & \text{if } a \neq x \\ P_T(G_a) \times \prod_{i=1}^{d} C_W(w_i), & \text{if } a = x, \end{cases}$$

$$P'_F(G_a) = \begin{cases} P_F(G_a), & \text{if } a \neq x \\ P_F(G_a) \times \prod_{i=1}^{d} (1 - C_W(w_i)), & \text{if } a = x. \end{cases}$$

  In those equations, $P'_T(G_a)$ ($P'_F(G_a)$) is the probability that all results in G_a and all d workers are correct (incorrect).

  The metric $EC_J(j)$ predicts the credibility of job j supposing that all d workers return the same result, which joins the result group G_x, where G_x is the result group which has a maximum credibility in all groups.

4.2.2. Job Scheduling Algorithm

Using metrics EC_J and ENR, we propose a new job scheduling method called "expected-credibility-based job scheduling". Figure **??** shows the algorithm of the expected-credibility-based job scheduling. First, all idle workers are stored in a worker queue (line 1 in Fig.**??**) and all unfinished jobs are grouped based on the value of ENR (line 2). Let S_i be the set of unfinished jobs of which ENR are equal to i. As long as the worker queue is not empty, the master extracts a worker from the queue and allocates a job to the worker (lines 6–23). When a worker w is extracted from the queue (line 8), by the spot-checking mechanism, a spotter job is allocated to w with spot-check rate q. One unfinished job is allocated to w if a spotter job is not allocated. In this case, the master creates a set of candidate jobs CS for job allocation (line 11). CS is a set of unfinished jobs of which ENR are less than $i + R$. Then, the master selects job j_x, which has a minimum $EC_J(j)$ in CS

```
1  Push idle workers in worker queue Q;
2  Group all unfinished jobs into S_i;  // S_i is a set of unfinished jobs of which ENR are i
3  i = 0;  // Select jobs from S_0 at first
4  CS = φ  // CS is a set of candidate jobs for job allocation
5
6  while ( Q is not empty ) do
7    while ( S_i != φ ) do
8      Pop worker w from Q;
9      Allocate a spotter job to worker w with rate q;
10     if( No job is allocated to w ) then
11       Create a set of candidate jobs CS = S_i + S_{i+1} + ... + S_{i+R}
12       Select job j_x which has a minimum EC_J in CS; // j_x ∈ S_x
13       Allocate job j_x to w;
14       Update EC_J(j_x);
15       S_x = S_x − {j_x};
16       S_{x+1} = S_{x+1} + {j_x};
17     end if
18     if ( Q is empty ) then
19       exit;
20     end if
21   end while
22   i = i + 1;
23 end while
```

Figure. 22: Expected-credibility-based job scheduling algorithm

(line 12). Note that j_x is a job in S_x, that is, $ENR(j_x) = x$. Once job j_x is allocated to w, then $EC_J(j_x)$, S_x, and S_{x+1} are updated immediately to reflect the allocation of job j_x in the subsequent scheduling (lines 14 – 16).

Using this scheduling method, jobs having smaller ENR are selected for candidate jobs CS prior to jobs having larger ENR. Also, among CS, a job having the minimum EC_J is selected prior to others. That is, jobs are selected in the ascending order of ENR and EC_J.

This scheduling policy arises from the careful observation that there exist many unnecessary job allocations to satisfy a job's termination condition: i.e. $C_J(j) \geq 1 - \epsilon_{acc}$. This occurs because C_J changes dynamically during a computation. In fact, $C_J(j)$ changes when the master (1) collects new results for job j, or (2) collects results for a spotter job from the workers which have returned results for job j. Regarding the second case, when a worker w survives spot-checking, $C_W(w)$ and $C_R(r)$ increase as in eqs. (8) – (10), which in turn increases the job's credibility $C_J(j)$. Because spot-checking is executed with a constant rate q, $C_J(j)$ tends to continue to increase during the computation, which implies that job j might satisfy the termination condition with no new results. Meanwhile, using a job scheduling mechanism, the master continues to allocate the unfinished job j and collects new results so that $C_J(j)$ satisfies the termination condition. If $C_J(j)$ exceeds the threshold without the newly collected results, then the new results will turn out to be unnecessary for job j. In such a case, such additional job allocations can be said to be "unnecessary job allocations".

The key factor to reduce the computation time T is how to save such unnecessary job allocations. It is, however, difficult to find during a computation which results will be unnecessary at the end of the computation because the credibility of jobs changes dynamically depending on the unpredictable results of jobs returned for normal and spotter jobs. Note that jobs which have lower credibility and fewer results require more results to increase the credibility; that is, to save unnecessary job allocations, a job which has lower EC_J

and ENR should be selected prior to others during job scheduling. If jobs are selected in the ascending order of EC_J without considering the number of results, the jobs for which erroneous results have been returned will be selected many times since the presence of incorrect results makes it more difficult to increase the jobs' credibility . Such job scheduling will cause performance degradation because of unnecessary job allocations, as verified in our performance evaluation presented in Sec. 4.3.2.. Therefore, as described above, the proposed job scheduling method selects jobs in the ascending order of ENR and EC_J.

4.2.3. Scheduling Cost

Next, the scheduling cost is analyzed for both the round-robin and our proposed methods. In VC systems, scheduling is the repetitive process of allocating jobs and receiving results. As described in this chapter, the scheduling cost is defined as the time (worker's waiting time) to select one job for a worker when the master receives a result from the worker.

Let t_c be the time required for the master to calculate a credibility of one job, C_J; also, t_e is the time to calculate the ENR of one job. The scheduling cost at time step t (t-th unit time) is denoted as $Cost(t)$. The scheduling cost varies according to when it is calculated. Therefore, we calculate $Cost(t)$ and mean of $Cost(t)$ for both methods.

In the round-robin method [23], when the master receives a result for a normal job j, the master simply calculates only $C_J(j)$ in t_c to check whether it reaches the threshold or not. If the master receives a result for a spotter job, the result will affect C_J of several jobs because of backtracking or updating of the credibility. The number of jobs which need the update of the credibility at time step t is, at most, t because the number of results produced by w is, at most, t even if w can return a result in every time step. The scheduling cost for the round-robin method, i.e. $Cost_{orig}(t)$, is calculated as

$$Cost_{orig}(t) = t_c((1-q) + qt). \tag{19}$$

Time step t changes from 1 to T, where T is the overall computation time of the computation represented in unit times. The average of $Cost_{orig}(t)$, i.e. $E(Cost_{orig})$ is given as

$$\begin{aligned} E(Cost_{orig}) &= \frac{1}{T}\sum_{t=1}^{T} Cost_{orig}(t) \\ &= t_c((1-q) + \frac{q(T+1)}{2}). \end{aligned} \tag{20}$$

Because the round-robin method is the simplest scheduling method, this cost is the minimum required for credibility-based voting.

In the proposed method, when the master receives a result for a normal job j, the master calculates $EC_J(j)$ and $ENR(j)$ in addition to $C_J(j)$. $EC_J(j)$ and $ENR(j)$ must also be updated when a normal job j is allocated to a worker (as shown in lines 13 – 16 in Fig. **??**). Therefore, for each allocation of a normal job (probability $1-q$), the master calculates $EC_J(j)$ and $ENR(j)$ twice. If the master receives a result for a spotter job, the master calculates EC_J and ENR in addition to C_J for, at most, t jobs as described above. The

scheduling cost for the proposed method, i.e. $Cost_{prop}(t)$, is calculated as

$$\begin{aligned} Cost_{prop}(t) &= (1-q)(t_c + 2(t_c + t_e)) + qt(t_c + (t_c + t_e)) \\ &= Cost_{orig}(t) + (t_c + t_e)(2(1-q) + qt). \end{aligned} \tag{21}$$

By definition, the time to calculate EC_J is almost identical as t_c (the time to calculate C_J).

The average of $Cost_{prop}(t)$, i.e. $E(Cost_{prop})$, is given as the following.

$$\begin{aligned} E(Cost_{prop}) &= \frac{1}{T}\sum_{t=1}^{T} Cost_{prop}(t) \\ &= 2 \times t_c((1-q) + \frac{q(T+1)}{2}) + t_c(1-q) \\ &\quad + t_e(2(1-q) + \frac{q(T+1)}{2}). \end{aligned} \tag{22}$$

The value of q is between 0 and 1 and $T >> 1$ (several orders of magnitude). Therefore, the second term in eq.(22) is negligibly small compared to the first. The third term in eq.(22) is also negligibly small compared to the first because calculation of ENR involves the simple addition shown in eq.(17), then t_e is negligibly small. Therefore, $E(Cost_{prop})$ can be represented as follows.

$$E(Cost_{prop}) \approx 2 \times E(Cost_{orig}). \tag{23}$$

From eqs. (20) and (23), it is apparent that the scheduling costs of both methods are negligibly small compared to the execution time of a job in general VC systems (e.g. several hours in [11]), because t_c, the calculation cost of one credibility, is a very small value (on the order of microseconds). Furthermore, both scheduling costs (for a single worker) are independent of the system scale, such as the number of jobs and the number of workers. Consequently, both methods will work well in large-scale VC systems, without a considerable increase in the workers' waiting time.

4.3. Performance Evaluation

4.3.1. Simulation Conditions

We evaluate the effectiveness of the proposed job scheduling method through the simulation of VCs. Computation times T and error rates ϵ of VCs are evaluated as the average of 100 simulation results for different job scheduling methods: random (denoted by "random"), round robin (denoted by "rr1") [23], and several variants of the proposed method. The variants of the proposed method select jobs using ENR and EC_J. They are called, respectively, "ENR method" and "EC_J method".

- ENR method:
  This method selects a job that has a minimum ENR in all unfinished jobs. A round-robin selection is used if some jobs have the same ENR.

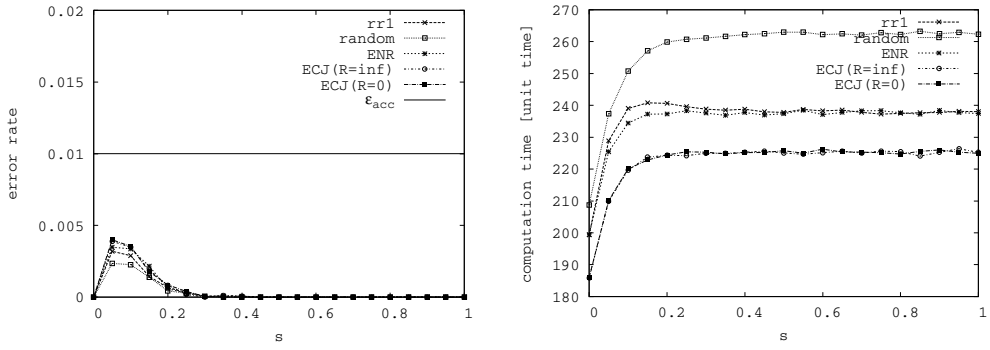
- EC_J method
  This method selects a job that has a minimum EC_J in the candidate jobs CS. Depending on the value of R, i.e. how to create CS, there are several variants of EC_J method.

– $R = 0$:
CS is a set of unfinished jobs of which ENR are minimum ($CS = S_i$). This method selects a job that has a minimum EC_J in minimum ENR jobs.

– $R = inf$:
CS is a set of all unfinished jobs ($CS = S_i + S_{i+1} + \ldots$). This method selects a job that has a minimum EC_J in all unfinished jobs.

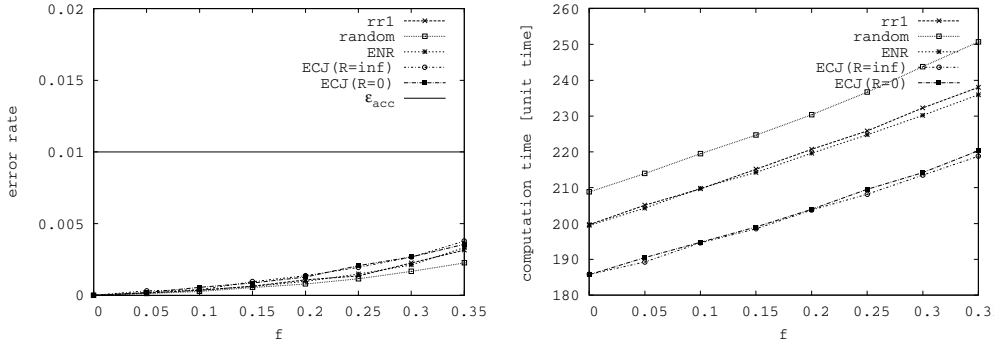
The parameters and assumptions in Sec. 3.2. are also used in simulations here.

4.3.2. Simulation Results



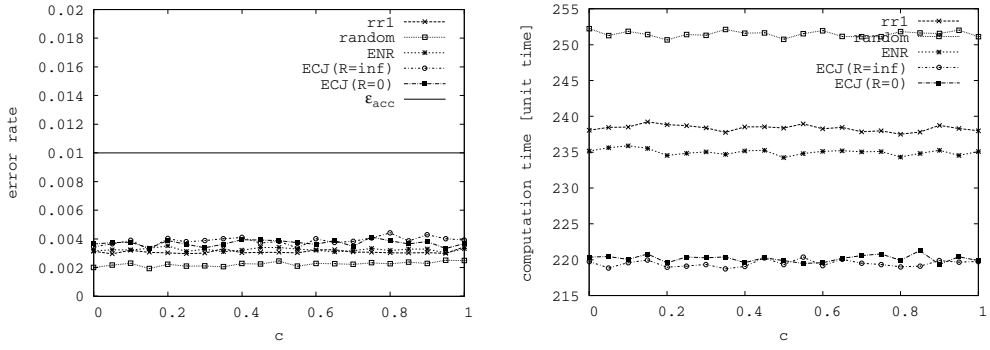
(a) Error-rate ϵ (b) Computation time T

Figure 23. Error rate and Computation time for sabotage rate s ($\epsilon_{acc} = 0.01$, $f = 0.35$, $c = 0.0$, $q = 0.1$, $p_d = 0$, with blacklisting)



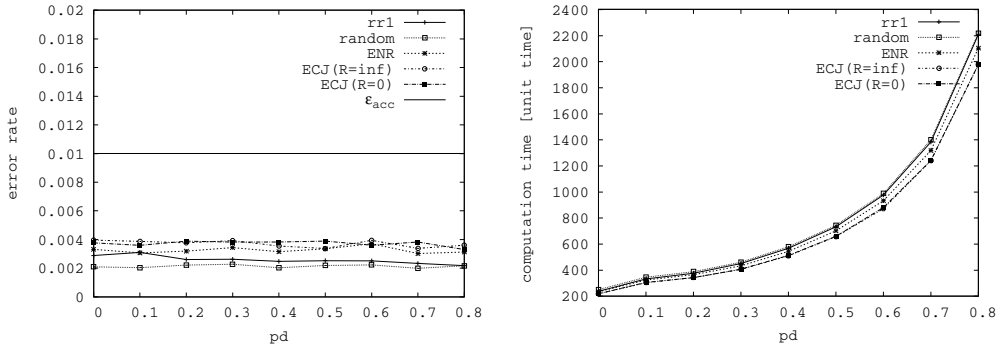
(a) Error-rate ϵ (b) Computation time T

Figure 24. Error rate and Computation time for fraction f ($\epsilon_{acc} = 0.01$, $s = 0.1$, $c = 0.0$, $q = 0.1$, $p_d = 0$, with blacklisting)



(a) Error-rate ϵ (b) Computation time T

Figure 25. Error rate and Computation time for colluding rate c ($\epsilon_{acc} = 0.01$, $s = 0.1$, $f = 0.35$, $q = 0.1$, $p_d = 0$, with blacklisting)



(a) Error-rate ϵ (b) Computation time T

Figure 26. Error rate and Computation time for defection rate pd ($\epsilon_{acc} = 0.01$, $s = 0.1$, $f = 0.35$, $q = 0.1$, $P_{up}(steady) = 0.8$, with blacklisting).

In cases with blacklisting Simulation results for VC systems with blacklisting are shown in Fig.23 – Fig.26.

Fig.23 (a) and Fig.24 (a) shows the error rates of each scheduling method as a function of sabotage rate s and fraction f, respectively. These figures show error rates of credibility-based voting are less than ϵ_{acc} regardless of the used scheduling method. This means credibility-based voting can guarantee the reliability condition $\epsilon \leq \epsilon_{eacc}$ for any s and f regardless of the used scheduling method.

Fig.23 (b) and Fig.24 (b) shows computation times of each scheduling method. As a function of sabotage rate s, the computation time of each method increases at first at s of 0.2, then retains almost identical values for larger s. This is true because, when s is large, all saboteurs are caught until the end of the computation and all results produced by the

saboteurs are invalidated, whether correct or not. Also, as f increases, the computation time of each method increases because the number of incorrect results returned from saboteurs increase proportionally to f.

These figures shows that the proposed EC_J method always outperforms the round-robin method for any s and f, even if saboteurs never return incorrect results ($s = 0$) or there are no saboteurs ($f = 0$). This is true because, even if there is no saboteur, different jobs have different credibility by the effect of spot-checking. Since proposed method dynamically selects suitable jobs based on EC_J, for any s and f, it performs well than round-robin method, which statically select jobs without considering credibility. This result means, although the actual values of s and f are unknown to the master, the proposed method can improve the computation time irrespective of the behavior of saboteurs s and the number of saboteurs f. This very important feature indicates that the proposed method is applicable to VC systems of various environments.

In addition, the performance curves of EC_J methods ($EC_J(R = inf)$ and $EC_J(R = 0)$) are close, whereas the curve of the ENR method is close to that of the round-robin method. This means the job scheduling method based only on the number of results, i.e. the ENR method, does not improve the computation time markedly because the credibility of jobs is different even though those jobs have the same ENR. To improve the computation time efficiently, a job scheduling method must take account of the job's credibility, as the proposed EC_J method does.

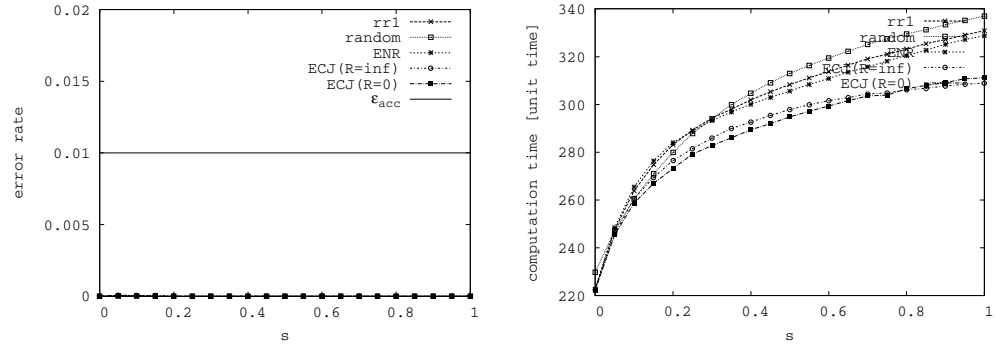
Fig.25 shows the error rate and computation time of each scheduling method as a function of colluding rate c. As shown in this figure, both error rate and computation time stay constant for all job scheduling methods. Also, there is a clear difference between the computation time of round robin method and that of EC_J method. This result indicates that EC_J method outperforms round robin method for any c, while guaranteeing the reliability condition $\epsilon \leq \epsilon_{acc}$.

Fig.26 shows the error rate and computation time of each scheduling method as a function of defection rate p_d. As shown in eq.(16), computation time is proportional to $1/(1 - p_d)$. This figure also shows that all job scheduling methods includes EC_J method guarantees the reliability condition for any p_d.

In cases without blacklisting Simulation results for VC systems without blacklisting are shown in Fig.27 – Fig.31.

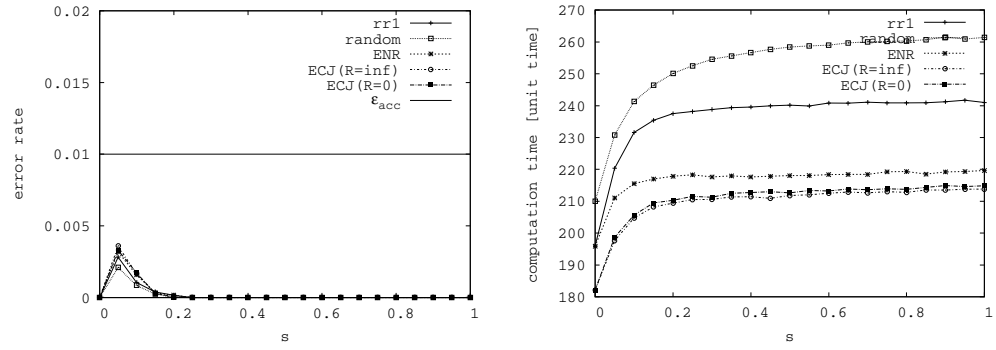
Fig.27 and Fig.28 show the error rate and the computation time as a function of sabotage rate s. Fig.29 shows the error rate and the computation time as a function of fraction f. These figures show that credibility-based voting can guarantee the reliability condition $\epsilon \leq \epsilon_{eacc}$ for any s and f, even in cases without blacklisting. These figures also show that the proposed EC_J method outperforms the round-robin method for any s and f, as in cases with blacklisting.

Fig.27 (b) shows computation time T as a function of sabotage rate s for $q = 0.1$. In cases without blacklisting, the sabotage rate s may affect the computation time, different from cases with blacklisting. This is true because the sabotage rate s is the ratio of incorrect results to all results produced by a saboteur. In cases without blacklisting, although a saboteur will be caught, the saboteur rejoins the system and produces incorrect results permanently. Some of produced incorrect results are retained in the system until the end of



(a) Error-rate ϵ (b) Computation time T

Figure 27. Error rate and Computation time for sabotage rate s ($\epsilon_{acc} = 0.01$, $f = 0.35$, $c = 0.0$, $q = 0.1$, $p_d = 0$, without blacklisting).

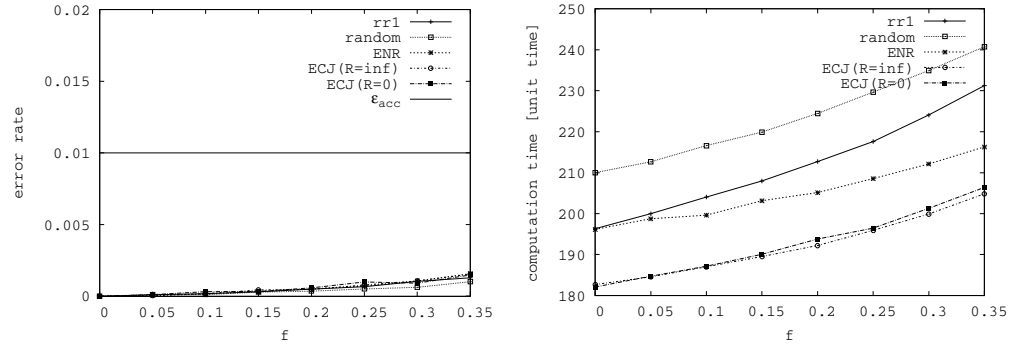


(a) Error-rate ϵ (b) Computation time T

Figure 28. Error rate and Computation time for sabotage rate s ($\epsilon_{acc} = 0.01$, $f = 0.35$, $c = 0.0$, $q = 0.2$, $p_d = 0$, without blacklisting).

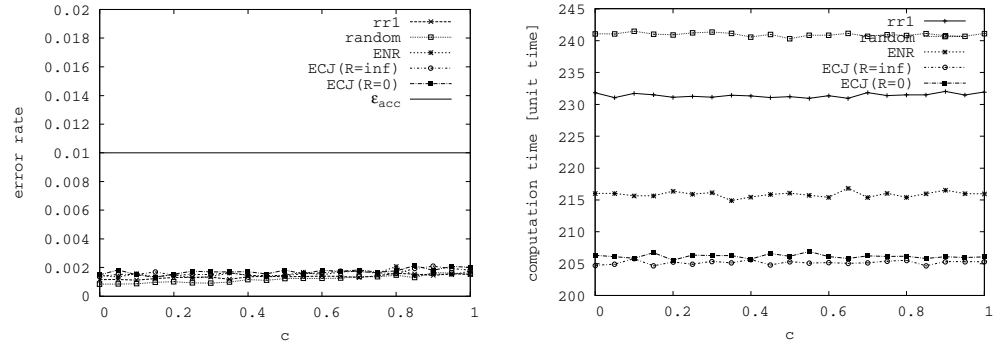
the computation, each of which makes it more difficult to increase the jobs' credibility. The number of these incorrect results is proportional to s. For that reason, the computation time may increase as s increases in cases without blacklisting.

Fig.27 (b) also shows that $EC_J(R = inf)$ method takes a longer computation time than $EC_J(R = 0)$ method when s is small.This is true because the $EC_J(R = inf)$ method does not take account of the number of results for each job. By selecting a job in the ascending order of EC_J, a job having incorrect results is selected prior to other jobs because the credibility of such a job is smaller, as shown in eqs. (11) – (14). On the other hand, when incorrect results are invalidated by backtracking, the credibility of such jobs becomes considerably larger; some results of the job will turn out to be unnecessary. Particularly, when s is small, it is difficult to catch saboteurs and invalidate incorrect results. Consequently,



(a) Error-rate ϵ (b) Computation time T

Figure 29. Error rate and Computation time for fraction f ($\epsilon_{acc} = 0.01$, $s = 0.1$, $c = 0.0$, $q = 0.2$, $p_d = 0$, without blacklisting).



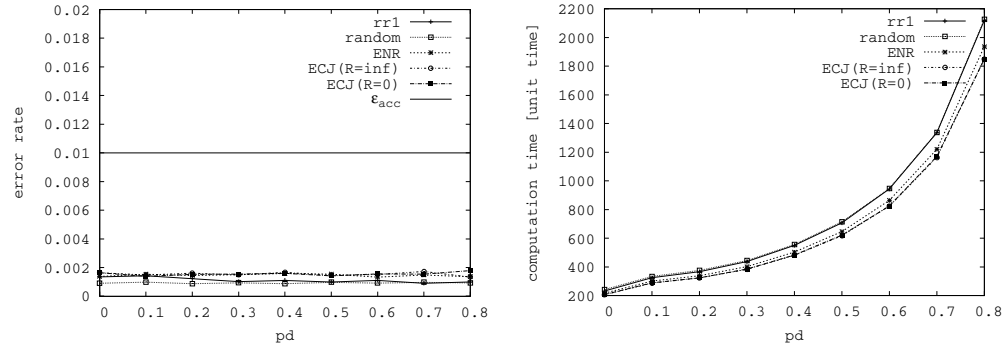
(a) Error-rate ϵ (b) Computation time T

Figure 30. Error rate and Computation time for colluding rate c ($\epsilon_{acc} = 0.01$, $s = 0.1$, $f = 0.35$, $c = 0.0$, $q = 0.2$, $p_d = 0$, without blacklisting).

jobs having incorrect results are selected repeatedly and the number of unnecessary job allocations increases. For that reason, $EC_J(R = inf)$ method requires a longer computation time, as shown in Fig.27 (b).

By selecting jobs in the ascending order of ENR, a job having numerous results is not selected in job scheduling even if the job has incorrect results. In this case, the unnecessary job allocations do not become numerous, as with the $EC_J(R = 0)$. Therefore, the variations of the proposed method $EC_J(R = 0)$, which selects jobs in the ascending order of ENR at first ($R = 0$) and then selects a job based on EC_J, shows better performance.

Fig.28 (b) shows computation time T as a function of sabotage rate s for $q = 0.2$. When the value of q becomes larger, all saboteurs are frequently detected by more spot-checking and no incorrect result remains in the system. In this case, the computation time



(a) Error-rate ϵ

(b) Computation time T

Figure 31. Error rate and Computation time for defection rate pd ($\epsilon_{acc} = 0.01$, $s = 0.1$, $f = 0.35$, $c = 0.0$, $q = 0.2$, $P_{up}(steady) = 0.8$, without blacklisting).

of credibility-based voting stays constant for any s even in cases without blacklisting.

Fig.28 (b) and Fig.29 (b) show that the difference between ENR method and the round-robin method is large, different from cases with blacklisting (Fig.24 (b)). In cases without blacklisting, the saboteurs can rejoin the system and produce incorrect results permanently, so that more results are invalidated by backtracking and each job's ENR becomes widely disproportionate. This result indicates that consideration of ENR becomes an important reason to select jobs in cases without blacklisting.

Fig.30 and Fig.31 show the error rate and computation time as functions of colluding rate c and defection rate p_d, respectively. As shown in these figures, error rates of all job scheduling methods stay constant for any c and p_d even in cases without blacklisting. These figures also show that all job scheduling methods includes EC_J method guarantees the reliability condition for any c and p_d. This result indicates that EC_J method outperforms round robin method for any c and p_d even in cases without blacklisting.

4.4. Conclusion

In this section, we proposed expected-credibility-based job scheduling for credibility-based voting [23] to improve the performance of the original round-robin job scheduling. The proposed job scheduling method selects jobs to save unnecessary job allocations based on two novel metrics: the expected-credibility and the expected number of results.

Through the extensive simulations, we found the following important results;

- Credibility-based voting can guarantee the reliability condition regardless of used job scheduling method.
- The proposed job scheduling method reduces the computation time compared to the round-robin method, irrespective of the value of unknown parameters s, f, c and p_d.

These very important features indicate that the proposed method is applicable to VC systems of various environments.

5. Conclusion

In this chapter, we discuss the sabotage-tolerance problems in VC systems which has been a mandatory issue for high performance and reliable VC environment.

In section 2, we introduced major VC projects/platforms and proposed some practical models of VC environment. In section 3, we introduced existing sabotage-tolerance mechanisms and provided their performance evaluation in the practical models. From this evaluation, we concluded that credibility-based voting is the most promising approach for sabotage-tolerance of VCs. This section also showed that the performance largely depends on the job scheduling method in credibility-based voting. Then, in section 4, we proposed expected-credibility-based job scheduling method for credibility-based voting. The extensive simulations showed that the proposed method reduced the computation time compared to the original round-robin method, irrespective of the value of unknown parameters.

In the process of applying the proposed job scheduling method to real VCs, some questions may arise with respect to assumptions or implementation details. Some promising research that can be further explored , for example, include:

- Workers' processor speed:
  All workers assumed to have the same property, e,g. the same speed, in VC models. Since each worker in real VC has different property, there may be a big difference between the simulation results and real environments. Additional simulation will show the actual performance in real VCs. For such simulation, we can still handle workers who have different property from others by simply treating them as multiple workers. For example, a worker with 2GHz processor can be assumed as two workers with 1GHz processors.

- Another form of collusion:
  The predefined colluding is assumed in the simulation. The predefined colluding method is the most simple model of collusion. Since saboteurs in this model always return incorrect results with constant probability s, it is easy to detect the collusion by spot-checking. Saboteurs in real VCs may behave more artfully, e,g, they collude only when they receive the same job each other since the collusion will succeed with a higher probability. Such saboteurs can be described using the direct colluding model.

References

[1] Luis F. G. Sarmenta, "Volunteer Computing", Ph.D. thesis, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, 2001.

[2] David M. Toth, "Improving the Productivity of Volunteer Computing", Ph.D. thesis, WORCESTER POLYTECHNIC INSTITUTE, 2008.

[3] Thomas E. Anderson, David E. Culler, David A. Patterson, and the NOW Team, "A case for networks of workstations: NOW", IEEE Micro, Feb. 1995.

[4] Scott Fields, “Hunting for Wasted Computing Power: New Software for Computing Networks Puts Idle PCfs to Work”, Research Sampler University of Wisconsin-Madison, 1993.

[5] Andrew S. Grimshaw, William A. Wulf, James C. French, Alfred C. Weaver, Paul F. Reynolds Jr., “A Synopsis of the Legion Project”, CS Technical Report CS-94-20, University of Virginia, June, 1994.

[6] I. Foster, and C. Kesselman, “Globus: A Metacomputing Infrastructure Toolkit”, *Intl J. Supercomputer Applications,* **11**(2) pp.115–128, 1997.

[7] H. Casanova and J. Dongarra, ‘NetSolve: A Network Server for Solving Computational Science Problems” Technical Report CS-95-313. University of Tennesee, 1995.

[8] Mitsuhisa Sato, Hidemoto Nakada, Satoshi Sekiguchi, Satoshi Matsuoka, Umpei Nagashima and Hiromitsu Takagi, “Ninf: A Network based Information Library for a Global World-Wide Computing Infrastructure”, in *Proc. of HPCN’97 (LNCS-1225)*, pp.491–502 , 1997.

[9] I. Foster, C. Kesselman, and S. Teucke, “The Anatomy of the Grid: Enabling Scalable Virtual Organizations”, *International J. Supercomputer Applications,* **15**(3), 2001.

[10] Gilles Fedak, CCcile Germain, Vincent NCri and Franck Cappello, “XtremWeb: A Generic Global Computing System”, *Proc. of the 1st IEEE International Symposium on Cluster Computing and the Grid*, pp. 582–587, 2001.

[11] SETI@home
http://setiathome.berkeley.edu/

[12] David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, Dan Werthimer, “SETI@home: an experiment in public-resource computing”, *Communications of the ACM,* Vol. 45, Issue 11, pp. 56–61, 2002.

[13] Great Internet Mersenne Prime Search GIMPS
http://www.mersenne.org/

[14] distributed.net
http://www.distributed.net/

[15] Einstein@Home
http://einstein.phys.uwm.edu/

[16] Rosetta@home
http://boinc.bakerlab.org/rosetta/

[17] BOINC
http://boinc.berkeley.edu/

[18] BOINC Statistics
http://boincstats.com/

[19] BOINC Computation credit
http://boinc.berkeley.edu/wiki/Computation_credit

[20] David P.Anderson, "BOINC: A System for Public-Resource Computing and Storage", *5th IEEE/ACM International Workshop on Grid Computing* , pp. 4–10, 2004.

[21] Bayanihan
http://www.cag.lcs.mit.edu/bayanihan/

[22] Luis F. G. Sarmenta, Satoshi Hirano, "Bayanihan: building and studying web-based volunteer computing systems using Java", *Future Generation Computer Systems* , Vol. 15, Issues 5–6, pp. 675–686. 1999.

[23] Luis F. G. Sarmenta, "Sabotage-Tolerance Mechanisms for Volunteer Computing Systems", *Future Generation Computer Systems* , Vol. 18, Issue 4, pp. 561–572, 2002.

[24] Patricio Domingues, Bruno Sousa, Luis Moura Silva, "Sabotage-tolerance and trust management in desktop grid computing", *Future Generation Computer Systems* , Vol. 23, Issue 7, pp. 904–912, 2007.

[25] Felipe Martins, Marcio Maia, Rossana M. de Castro Andrade, Aldri L. dos Santos, Jose Neuman de Souza, "Detecting Malicious Manipulation in Grid Environments", *The 18th International Symposium on Computer Architecture and High Performance Computing*, pp. 28–35, 2006.

[26] Shanyu Zhao, Virginia Lo, Chris GauthierDickey, "Result Verification and Trust-Based Scheduling in Peer-to-Peer Grids", *The 5th IEEE International Conference on Peer-to-Peer Computing* , pp.31 – 38, 2005

[27] Derrick Kondo, Filipe Araujo, Paul Malecot, Patricio Domingues, Luis Moura Silva, Gilles Fedak, Franck Cappello, "Characterizing Error Rates in Internet Desktop Grids", *13th European Conference on Parallel and Distributed Computing,* pp. 361–371, 2007.

[28] Wenliang Du, Jing Jia, Manish Mangal, Mummoorthy Murugesan, "Uncheatable grid computing", *The 24th International Conference on Distributed Computing Systems,* pp. 4 –11, 2004

[29] Jason Sonnek, Abhishek Chandra, Jon Weissman, "Adaptive Reputation-Based Scheduling on Unreliable Distributed Infrastructures", *IEEE Transactions on Parallel and Distributed Systems* , Vol. 18, No. 11, pp. 1551–1564, 2007.

[30] Baohua Wei, Fedak, G., Cappello, F., "Collaborative Data Distribution with BitTorrent for Computational Desktop Grids", *The 4th International Symposium on Parallel and Distributed Computing,* pp. 250 – 257, 2005

[31] Kim J.-S., Nam B., Marsh M., Keleher P., Bhattacharjee B., Richardson D., Wellnitz D., Sussman A. "Creating a Robust Desktop Grid using Peer-to-Peer Services", *The 21st IEEE International Parallel and Distributed Processing Symposium IPDPS 2007,* pp. 1 – 7, 2007

[32] Silaghi G., Araujo F., Domingues P., Silva L. M., Arenas A., "Defeating Colluding Nodes in Desktop Grid Computing Platforms", *The 22nd IEEE International Parallel and Distributed Processing Symposium IPDPS 2008,* PCGRID 2008 workshop, pp. 1 – 8, 2008

[33] Failure Trace Archive
http://fta.inria.fr/

[34] Derrick Kondo, Bahman Javadi, Alex Iosup, Dick Epema, "The Failure Trace Archive: Enabling Comparative Analysis of Failures in Diverse Distributed Systems", *INRIA technical report number 00433523* , Nov. 2009.

[35] Derrick Kondo, Gilles Fedak, Franck Cappello, Andrew A. Chien, Henri Casanova, "Characterizing resource availability in enterprise desktop grids", *Future Generation Computer Systems,* Vol. 23, Issue 7, pp. 888–903, 2007.

[36] Yu. A. Zuev, "On the Estimation of Efficiency of Voting Procedures", *Theory of Probability and its Applications,* Vol. 42, Issue 1, pp. 71–81, 1998.

[37] Java Platform Documentation (Standard Ed. 6)
http://java.sun.com/javase/6/docs/api/java/util/Random.html

[38] Kan Watanabe, Masaru Fukushi, Susumu Horiguchi, "Optimal Spot-checking to minimize the Computation Time in Volunteer Computing", The 22nd IEEE International Parallel and Distributed Processing Symposium IPDPS 2008, PCGRID 2008 workshop, pp. 1 – 8, 2008

[39] Kan Watanabe, Masaru Fukushi and Susumu. Horiguchi, "Optimal Spot-checking for Computation Time Minimization in Volunteer Computing", *Journal of Grid Computing,* Vol. 7, Issue 4, pp.575 - 600, 2009

[40] Kan Watanabe, Masaru Fukushi, Susumu Horiguchi, "Expected-credibility-based Job Scheduling for Reliable Volunteer Computing", *IEICE Transactions on Information and Systems*, Vol.E93-D, No.2, pp. 306–314, 2010.

In: Computer Science Research and Technology. Volume 2 ISBN 978-1-61122-073-5
Editor: Janice P. Bauer

Chapter 8

SCALABLE PACKET CLASSIFICATION BY USING HASH-BASED ALGORITHMS

Pi-Chung Wang
Department of Computer Science and Engineering,
National Chung Hsing University, Taichung, Taiwan

Abstract

In next-generation networks, packet classification is used to categorize incoming packets into multiple forwarding classes based on pre-defined filters and make information accessible for quality of service or security handling in the network. In the last decade, the technique of packet classification has been widely deployed in various network devices, including routers, firewalls and network intrusion detection systems. To pursue better search performance, numerous algorithms of packet classification have been proposed and optimized for certain filter databases; however, these algorithms may not scale in either storage or speed performance or both for the filter databases with different characteristics. Specifically, packet classification with a large number of filters usually exhibits poor worst-case performance.

In this chapter, we improve the performance of packet classification by using multiple hash tables. The existing hash-based algorithms have superior scalability with respect to the required space; however, their search performance may not be comparable to other algorithms. We observe the characteristics of the hash-based algorithms and introduce three hash-based algorithms to improve the performance of packet classification. In the first algorithm, we combine two complementary algorithms, Cross-producting and Pruned Tuple Space Search, to make packet classification both fast and scalable. Unlike the existing algorithms whose performance is contingent on the filter database attributes, our algorithm shows better scalability and feasibility. We also introduce the procedure of incremental updates. Next, we improve the performance of Pruned Tuple Space Search by combining the Aggregate Bit Vector algorithm. We also show the related update procedures. Last, we present a hash-table reordering algorithm to minimize the number of accessed hash tables with the aid of bitmaps. We also use pre-computation to ensure the accuracy of our search procedure. For each algorithm, we evaluate the performance with both real and synthetic filter databases of varying sizes and characteristics. The experimental results demonstrate that the new algorithms improve the speed and storage performance simultaneously. Moreover, the results also demonstrate that our schemes are feasible and scalable.

1. Introduction

In next generation networks (NGNs), packet classification is important in fulfilling the requirements of differentiated services. By using pre-defined filters, packet classification could categorize an incoming packet to a forwarding class, such as those indicated by the differentiated services codepoint (DSCP field) in the **DiffServ** model [1]. It has been extensively employed in the Internet for secure filtering and service differentiation. Various devices, such as routers, firewalls and network intrusion detection systems, have used packet classification to practical effect. Therefore, the performance of packet classification is important in the deployment of NGNs. However, packet classification with a potentially large number of filters is complex and exhibits poor worst-case performance [2, 3].

To perform packet classification, network devices would need a database of header filters with two or five fields. The value in each field could be a variable-length prefix, range, explicit value or wildcard [4]. The fields include the source/destination IP address prefix, source/destination port range of the transport protocol and the protocol type in the packet header. The number of fields used to inspect packet headers is determined by the particular application of packet classification. For example, packet classification with two-field filters is useful for QoS packet forwarding with traffic-engineered source-destination paths and multicasting [5], whereas with the five-field filters it can be applied to flow-based traffic engineering or network security [4]. Since two-field filters can be handled by packet classification with five-field filters, we focus on the latter in this paper. In a packet classifier, each filter is associated with an action, and each action is assigned a cost. Formally, we define a filter F with *d* fields as $F = (f_1, f_2, \ldots, f_d)$. A packet header P is said to match a particular filter F if for all i, the i_{th} field of the header satisfies f_i. The packet classification problem is to determine to which action each packet should take by inspecting the header fields of the packet and comparing them to a list of filters.

The problem of packet classification resembles point location problems in the multi-dimensional space [2, 3]. The point location problem is a set-membership problem that finds the enclosed region of a point within a set of non-overlapping regions. The best upper bounds classifying packets on N filters with *d* fields are either O($\log N$) time and O(N^d) space or O($\log^{d-1} N$) time and O(N) space. In other words, the performance would be totally unacceptable in the worst case. For instance, assume we attempt to process 1,024 filters of *four* fields in a router. An algorithm with O($\log^{d-1} N$) execution time and O(N) space requires 1,000 memory accesses per packet, which is impractical for network applications. If we use an O($\log N$) time and O(N^d) space algorithm, the space requirement becomes unrealistically large in the range of *one* terabyte. Furthermore, none of the above algorithms addresses the problem of arbitrary overlaps.

Much effort has been devoted to packet classification in recent years, yielding a number of algorithms. However, the existing algorithms may not scale well for large filter databases and sometimes a tradeoff between space and time is inevitable. For example, the trie-based algorithms [6, 7] and hash-based algorithms [8] have superior storage performance, but their search performance is limited. While the decision-tree-based algorithms [9, 10] and the combination-based algorithms [2, 6] have excellent search performance, they cannot support filter databases with numerous *wildcards* or *distinct field specifications*. Ternary CAMs (TCAMs) have superior performance in both space and time performance, but the capac-

ity limitation and power consumption make TCAMs less scalable for large filter databases. Therefore, new algorithmic schemes that can optimize the space-time tradeoff of the existing schemes by an order of magnitude are still attractive.

In this chapter, we present three techniques for improving the performance of packet classification by using multiple hash tables. The existing hash-based algorithms have superior scalability with respect to the required space; however, their search performance may not be comparable to other algorithms. We observe the characteristics of the hash-based algorithms and notice that the required storage of *Pruned Tuple Space Search*, a notable hash-based algorithm, is linearly proportional to the number of filters, but it cannot effectively bound the number of accessed hash tables. Therefore, we introduce three techniques to improve the performance of hash-based packet classification.

In the first algorithm, we combine two complementary algorithms, Cross-producting and Pruned Tuple Space Search, to improve the performance of packet classification. *Cross-producting* has superior search performance but needs exponentially increasing storage requirements in the worst case. Thus, both algorithms are complementary to each other. We design a novel combination based on the characteristics of both algorithms. We store the filters which would severely increase the storage requirement of *Cross-producting* in the data structure of *Pruned Tuple Space Search* while keeping the number of hash tables as low as possible. Unlike the existing algorithms whose performance is contingent on the filter database attributes, our algorithm shows better scalability and feasibility. We also introduce the procedure of incremental updates.

Next, we use the Aggregate Bit Vector algorithm to improve the performance of the Pruned Tuple Space Search algorithm. While both algorithms may not scale well for certain filter databases, they are also functionally complementary to each other. Thus, we store the filters into different data structures according to their characteristics. This scheme also reduces the number of hash tables with the minimal number of bit vectors. As a result, the performance of incremental update is improved as well as the search efficiency.

Last, we present a hash-table reordering algorithm to minimize the number of accessed hash tables with the aid of bitmaps. By reordering the hash tables, the search performance could be significantly improved while keeping the benefits from *Pruned Tuple Space Search*. We also use pre-computation to ensure the accuracy of our search procedure.

For each algorithm, we evaluate the performance with both real and synthetic filter databases of varying sizes and characteristics. The experimental results demonstrate that the new algorithms improve the speed and storage performance simultaneously. Moreover, the results also demonstrate that our schemes are feasible and scalable.

2. Related Work

We provide a brief discussion of the existing algorithms by dividing them into two categories according to their methods for classifying packets. The algorithms of the first category process the multidimensional filters directly; hence, packet classification is carried out upon the single multidimensional data structure. The algorithms of the other category decompose the problem of multidimensional packet classification into multiple one-dimensional cases. These algorithms start their search procedures from performing one-

dimensional searches upon individual fields and then combine their results. The following discussion will introduce some of the algorithms from both categories.

Let us begin with the algorithms of the first category whose search procedures do not perform one-dimensional searches. Ternary content addressable memories (TCAMs) are specialized memories which could store entries with arbitrary bit mask by storing an extra "Don't Care" in each cell. Then, TCAMs compare search data against the stored filters and return the address of the first matching filter. TCAMs have been proven effective for packet classification with a high degree of parallelism [11]. The drawbacks of TCAMs include their smaller density, power dissipation and extra entries due to the range-to-prefix transformation [7, 12]. Moreover, TCAMs with a particular word width cannot be used when a flexible filter specification is required. It is also quite difficult to produce TCAMs with large word widths to fill all bits in a filter. Much effort has gone into improving the power and storage efficiency of TCAMs [13–15]. The algorithms presented in [9] and [16] are based on decision trees. Both schemes attempt to divide the filters into multiple linear search buckets by using the data structure of a decision tree. The cut rules of filter categorization may either be a value [9] or a bit [16] of any field. *HyperCuts* [10] is a well-known scheme that adopts multidimensional cut rules and whose performance is better than the decision-trees-based algorithms with single-dimensional cut rules. Another algorithm is *Extended Grid-of-Tries with Path Compression* (*EGT-PC*) evolved from *Grid-of-Tries* (*GT*) in [6] that supports filters with more than two fields [7] with a series of linear searches. In [17] and [18], another two trie-based algorithms which are focused on hardware-based improvements have been proposed. The tuple-based algorithms store the filters with an identical prefix length combination in a hash table [8]. The prefixes can be concatenated to create a hash key for a hash table access. The matching filters can be found by probing each hash table alternately. Each hash table is also known as a tuple, and tuple space search aims at searching for the matching filters in the set of tuples. Two well-known tuple-based algorithms, *Rectangle Search* [8] and *Entry Pruned Tuple Search* (*EPTS*) [19], are designed to improve the performance of tuple space search.

Among the algorithms of the second category, the *Bit Vector* (*BV*) algorithm [20] performs *d* one-dimensional searches to derive *d* lists of filters with at least one matching field. By intersecting d filter lists, we could yield the lists of matching filters. Since the filter list is in the form of bit vectors, the *BV* algorithm is suitable for hardware implementation. A later work, *Aggregate Bit Vector* [12], has demonstrated dramatic improvement in the speed performance of the *BV* algorithm. Srinivasan et al. [6] introduced *Cross-producting* which uses best matching prefix lookups and a pre-computed table to combine the results of individual prefix lookups in each header field. Gupta and McKeown presented *Recursive Flow Classification* that does cross-producting recursively and can be viewed as a generalization of cross-producting [2]. In each iteration, only a subset of the inspected fields is used to generate a cross-product table and the unmatched entries are eliminated for storage saving. *Independent Sets* also supports multidimensional search. The idea behind *Independent Sets* is to categorize multidimensional filters according to the specifications of one field [21]. The filters in each independent set are mutually disjoint; therefore, binary search can be used to derive the matching filter in each independent set. The number of independent sets thereby determines the search performance of the algorithm. In [22], *Controlled Cross-producting* are proposed by combining *Cross-producting* and *Independent Sets*. However,

to support filter updates, *Controlled Cross-producting* must use hash-based cross-product tables. In addition, the update performance of *Controlled Cross-producting* in the worst case cannot support incremental updates. Several tuple-based algorithms also rely on one-dimensional searches for speed-up, such as *Pruned Tuple Space Search* [8].

Among the existing algorithms, the algorithms based on decision trees [9, 10] and cross-producting [2, 6] have the best search performance, while *Pruned Tuple Space Search* [8] and *Independent Sets* [21] have the least storage requirements. However, the existing algorithms may not scale well for large filter databases and sometimes a tradeoff between time and storage is inevitable. For example, *Pruned Tuple Space Search* [8] and *Independent Sets* [11] do not seem to include a bound on search latency, and the schemes in [2, 6] may not be updatable and scalable in the size of filter databases [4, 11]. Therefore, we present several new ideas to increase the storage efficiency of filter data structures as well as the scalability of packet classification.

3. Hashing and Cross-Producting

3.1. Motivation

From the above analysis of the existing algorithms, we notice that those based on *Cross-producting* usually have superior speed performance. However, these algorithms usually suffer from the problem of high memory requirements. For example, the algorithm presented in [6] only supports up to several hundreds of filters and other algorithms presented in [2] and [23] extend the number of supported filters to several thousands. These algorithms may not be feasible for the applications of the next generation networks due to the requirement of massive filters. To improve the feasibility of *Cross-producting*, we propose a novel approach to be combined with tuple space search to promote storage efficiency. As compared to the original *Cross-producting*, our approach also has other advantages, including the support of storage adjustment and incremental updates.

Before introducing our scheme, we will discuss the cross-producting-based algorithms in terms of their important properties that lead to the inspiration of the proposed algorithm. Srinivasan et al. introduced the seminal technique of cross-producting [6]. The *Cross-producting* algorithm is motivated by the observation that the number of unique field specifications is usually significantly less than the number of filters in the filter databases. Therefore, the *Cross-producting* algorithm pre-computes the best matching filter for each combination of the d field specifications in the cross-product table. To classify packets, the algorithm does d one-dimensional searches to find the best matching prefix of each field. Next, it retrieves the combination corresponding to the best matching prefixes to derive the index of the best matching filter from the cross-product table.

We use a set of eleven two-field filters as an example to further illustrate the cross-producting algorithm. As shown in Table 1, the field f_1 contains eight unique prefixes which are labeled from $0 \sim 7$, respectively. Likewise, the seven unique prefixes of the second field are also labeled from $0 \sim 6$. A cross-product table based on a direct-access array will contain $8 \times 7 = 56$ entries. These 56 entries include pseudo filters and empty filters. For instance, the first and second entries are pseudo filters and the third and fourth entries are empty filters. Among these pseudo filters, there are eleven entries corresponding

to the original filters and the rest of the entries are generated due to the process of cross-product. Note that the pseudo filters could match more than one original filter, such as the 9^{th} entry in the cross-product table.

Table 1. An example with eleven filters on two fields as well as their distinct field specifications and cross-product table.

Filter	f_1	f_2
F_1	$*$	0010
F_2	$*$	0111
F_3	$0*$	$1*$
F_4	$0*$	$01*$
F_5	$00*$	$11*$
F_6	$000*$	$01*$
F_7	$111*$	$11*$
F_8	$101*$	$01*$
F_9	$111*$	$00*$
F_{10}	0111	$*$
F_{11}	1111	$*$

No.	f_1	f_2
0	$*$	0010
1	$0*$	0111
2	$00*$	$1*$
3	$000*$	$01*$
4	$111*$	$11*$
5	$101*$	$00*$
6	0111	$*$
7	1111	

No.	Cross-product	Matching Filters
0	$(*, 0010)$	F_1
1	$(*, 0111)$	F_2
2	$(*, 1*)$	*none*
3	$(*, 01*)$	*none*
•	•	•
•	•	•
6	$(*, *)$	*none*
7	$(0*, 0010)$	F_1
8	$(0*, 0111)$	F_2, F_4
•	•	•
•	•	•
54	$(1111, 00*)$	F_{11}
55	$(1111, *)$	F_{11}

The process of packet classification is described as follows. Assume that we get a packet with a header value "0001,1110". We perform the longest prefix matching on each field and find the best matching prefixes, $000*$ and $11*$, whose indices are 3 and 4, respectively. The $25^{th}(= 3 \times 7 + 4)$ cross-product entry is retrieved and the matching filters are F_3 and F_5.

We can adopt a hash-based cross-product table to improve the storage performance of *Cross-producting*. In [6] and [2], the cross-product table is assumed to be implemented with a direct access array, in which each combination corresponds to one entry of the table. However, there are usually several combinations that do not match any filter. Consider a filter database with two filters: $\langle W, X\rangle$ and $\langle Y, Z\rangle$. The ranges of W and Y are disjointed, so do those of X and Z. There are four entries in the cross-product table, and two of these entries, $\langle W, Z\rangle$ and $\langle Y, X\rangle$, do not match any filters. Therefore, these two entries can be removed. To remove these useless entries, we can simply adopt a hash table to store those cross-product entries which match at least one filter. For those header values which do not match any cross-product entries in the hash table, we simply report that there is no matching filter. Nevertheless, such approach would incur possible hash collisions to degrade search performance in the worst case. Another algorithm, *Recursive Flow Classification* (*RFC*) [2], eliminates unmatched entries by gradually cross-producting a subset of the inspected fields. Although *RFC* could avoid the problem of hash collisions, it incurs repeatedly table accesses for each packet classification. Also, for the two-field filters, *RFC* is identical to *Cross-producting*. Moreover, in the case without any unmatched entries, *RFC* consumes more storage than *Cross-producting*. In a recent work, *Distributed Crossproducting of Field Labels* ($DCFL$) [23] also attempts to remove the unnecessary entries. It also improves the representation of crossproducting entries in [2] by using efficient set membership data structures and results in a much smaller storage, although it heavily relies on hardware

parallelism.

Unlike previous approaches that focus on removing unnecessary entries, our algorithm attempts to reduce the number of overall cross-product entries. Our idea is inspired by the correlation between the number of distinct field specifications and the required storage of the cross-product table. While storage performance depends on the number of distinct field specifications, the cross-producting-based schemes would suffer from severe degradation on storage performance when the size of filter database expands. Nevertheless, there are some algorithms that perform well on the filter databases with a very large number of filters, such as the tuple-based and decision-tree-based algorithms. However, the decision-tree-based algorithms have the problem of filter replications and result in poor storage performance in the worst case [24]. Hence, we use tuple-based algorithms to improve the storage performance of *Cross-producting*.

With tuple space search, the filters are categorized into hash tables according to their prefix length combinations. The filters with an identical length combination are stored and accessed by using a hash function. For example, the two-field filters, $\langle 000*, 01* \rangle$ and $\langle 111*, 11* \rangle$, both belong to the tuple $T_{3,2}$. When searching for $T_{3,2}$, a hash key is constructed by concatenating three bits of f_1 with two bits of f_2. Since the filters in the same tuple can be accessed with one hash probe, tuple space search is suitable for a large amount of filters with one or a few length combinations, which results in fewer tuples.

To use tuple space search, we need to transform the range fields of the original filters into prefixes since the idea of tuple space is to store filters in different hash tables according to the prefix-length combinations. The range fields are split into multiple subranges, where each subrange uniquely corresponds to one prefix [8]. For example, the range from 1024 to 65535 can be represented by *six* prefixes: "$000001*$", "$00001*$", "$0001*$", "$001*$", "$01*$" and "$1*$". Once a range is converted into more than one prefix, the corresponding filter is duplicated as well.

3.2. Algorithm

Based on the property of tuple space search, we divide the filters into two categories, where one category is searched by *Cross-producting* and the other is searched by tuple space search. Our algorithm starts from selecting filters that are suitable for tuple space search. We describe the filter categorization procedure of our algorithm in the following.

First, let us define a threshold value, $C_{threshold}$, to limit the number of cross-product entries. The threshold value can be set to the size of on-chip static random access memory (SRAM). By storing the cross-product table in SRAM, the latency of accessing cross-product tables can be shortened. Next, we derive the original number of cross-product entries by multiplying the numbers of distinct prefix specifications of each field. We denote the number of distinct prefixes of the i^{th} field as P_i. Then, the number of cross-product entries can be expressed as $C = \prod_{i=1}^{d} P_i$. In the third step, we construct a tuple space according to the length combinations of filters.

Assume that the number of the distinct field specifications of the i^{th} field in tuple T_j is denoted as $P_i^{T_j}$. The number of cross-product entries for the filters in T_j can be denoted as $C_{T_j} = \prod_{i=1}^{d} P_i^{T_j}$. However, to estimate the number of cross-product entries for those filters out of T_j, we need an extra value, $unique(P_i^{T_j})$, to denote the number

of distinct and unique prefixes in T_j, where the *unique* prefixes only appear in T_j. With $unique(P_i^{T_j})$, we can derive the number of cross-product entries for the filters not in T_j as $C' = \prod_{i=1}^{d}(P_i - unique(P_i^{T_j}))$. Since our algorithm attempts to reduce the value of C' to less than $C_{threshold}$, in the fourth step, we repeatedly select one tuple which can yield the minimum C' value until $C' \leq C_{threshold}$. In each iteration, we update the value of P_i as $P_i = P_i - unique(P_i^{T_j})$. Also, the number of unique prefixes of each tuple might change due to the fact that the other filters that specify the same prefixes belong to the tuples which have been selected in the previous iterations.

Note that our procedure of selecting tuples may not be optimal since we cannot calculate all combinations of tuple selections. The number of tuple selection combinations is equal to $\begin{pmatrix} |TS| \\ 1 \end{pmatrix} + \begin{pmatrix} |TS| \\ 2 \end{pmatrix} + \ldots + \begin{pmatrix} |TS| \\ |TS|-1 \end{pmatrix}$, where $|TS|$ is the number of original tuples. The problem of tuple selection is analogous to the facility location problem and can be modeled as a p-median problem [25]. This model is also known as a knapsack model, which is NP hard [25]. Therefore, we employ a greedy algorithm to simplify the calculation. Our experimental results demonstrate that the proposed algorithm could effectively reduce the cross-product entries while maintaining the number of tuples low.

We use the filters in Table 1 as an example to illustrate our procedure for tuple selection. We assume that $C_{threshold}$ equals *ten*. Table 2∼4 lists three iterations of calculating the number of the resulted cross-product entries. Initially, the value of P_1 is *eight* and that of P_2 is *seven*. Therefore, the number of cross-product entries, C, is equal to 56. In the first iteration, selecting the filters from $T_{3,2}$ would reduce the maximum number of cross-product entries, where there are 30 entries left in the cross-product table. After removing the filters in $T_{3,2}$, the values of P_1 and P_2 are then updated to *five* and *six*, respectively. Since the value of C' is 30, another iteration of tuple selection is activated.

Table 2. The First Iteration of Tuple Selection for the Filters in Table 1. ($P_1 = 8, P_2 = 7$)

Tuple	Filters	$unique(P_1^T)$	$unique(P_2^T)$	$P_1 - unique(P_1^T)$	$P_2 - unique(P_2^T)$
$T_{0,4}$	F_1, F_2	1	2	7	5
$T_{1,1}$	F_3	0	1	8	6
$T_{1,2}$	F_4	0	0	8	7
$T_{2,2}$	F_5	1	0	7	7
$T_{3,2}$	$F_6 \sim F_9$	3	1	5	6
$T_{4,0}$	F_{10}, F_{11}	2	1	6	6

In the second iteration, the values of $unique(P_1^T)$ and $unique(P_2^T)$ for the remaining tuples are updated. For example, the value of $unique(P_2^{T_{1,2}})$ is changed from 0 to 1 since the other two filters, F_6 and F_8, which specify "$01*$" on f_2, both belong to $T_{3,2}$ and have been removed in the first iteration. Our algorithm would select $T_{4,0}$ in this iteration, and the values of P_1 and P_2 are then updated to *three* and *five*, respectively. The resulted value of C' is equal to 15; thus, the third iteration of tuple selection is required. After updating the values of $unique(P_1^T)$ and $unique(P_2^T)$, $T_{0,4}$ is selected and the number of cross-product entries is reduced to *six* which is smaller than $C_{threshold}$. As a result, there remain three

tuples, $T_{0,4}, T_{3,2}$ and $T_{4,0}$, in the tuple space and the filters in the other tuples, F_3, F_4 and F_5, are inserted into the cross-product table, as listed in Table 5.

Table 3. The Second Iteration of Tuple Selection for the Filters in Table 1. ($P_1 = 5, P_2 = 6$)

Tuple	Filters	$unique(P_1^T)$	$unique(P_2^T)$	$P_1 - unique(P_1^T)$	$P_2 - unique(P_2^T)$
$T_{0,4}$	F_1, F_2	1	2	4	4
$T_{1,1}$	F_3	0	1	5	5
$T_{1,2}$	F_4	0	1	5	5
$T_{2,2}$	F_5	1	1	4	5
$T_{4,0}$	F_{10}, F_{11}	2	1	3	5

Table 4. The Third Iteration of Tuple Selection for the Filters in Table 1. ($P_1 = 3, P_2 = 5$)

Tuple	Filters	$unique(P_1^T)$	$unique(P_2^T)$	$P_1 - unique(P_1^T)$	$P_2 - unique(P_2^T)$
$T_{0,4}$	F_1, F_2	1	2	2	3
$T_{1,1}$	F_3	0	1	3	4
$T_{1,2}$	F_4	0	1	3	4
$T_{2,2}$	F_5	1	1	2	4

Table 5. The Resulted Data Structures for the Filters in Table 1.

Tuple Space	
Tuple	Filters
$T_{0,4}$	F_1, F_2
$T_{3,2}$	$F_6 \sim F_9$
$T_{4,0}$	F_{10}, F_{11}

Cross-product Table (F_3, F_4, F_5)		
No.	Cross-product	Matching Filters
0	$(0*, 1*)$	F_3
1	$(0*, 01*)$	F_4
2	$(0*, 11*)$	*none*
3	$(00*, 1*)$	F_3
4	$(00*, 01*)$	F_4
5	$(00*, 11*)$	F_3, F_5

After selecting the filters for tuple space search, the left filters are inserted into a cross-product table. The prefixes specified in the filters in the cross-product table are numbered and inserted into d data structures of best matching prefixes. For each prefix specification combination, the best matching filter is calculated and inserted into the cross-product table. To decide the best matching filter in the cross-product table, d one-dimensional searches for the best matching prefixes are executed first and the corresponding combination of prefix specifications is retrieved to derive the best matching filter.

We use *Pruned Tuple Space Search* to search for the other filters. *Pruned Tuple Space Search* is an advanced algorithm that eliminates tuples which do not match a query. For each field, the matching tuple lists of all prefixes are collected in d pruning tables. Accordingly, the lookup procedure starts by searching the best matching entries in the pruning tables. By intersecting d tuple lists of all fields, the tuples without matching filters are likely *pruned*. Although there are other algorithms which could support packet classifica-

tion with more than two fields, *Pruned Tuple Space Search* has several advantages that make it suitable to combine with *Cross-producting*. First, *Pruned Tuple Space Search* does not need complex pre-computation; thus, it could support incremental updates. Second, since no extra entry is required except for the pruning tables, *Pruned Tuple Space Search* has low memory consumption and update cost. Last, both algorithms perform one-dimensional searches; hence, we can merge these two algorithms seamlessly to achieve better search performance. To perform *Pruned Tuple Space Search*, we insert the specified prefixes into the one-dimensional data structure along with the tuple pruning information for each field. Each prefix might appear in the cross-product table, the tuple space, or both. In the former two cases, the prefix that only appears in one data structure must specify its longest sub-prefix in another data structure. As a result, each one-dimensional search can derive two best matching prefixes, *one* for *Cross-producting* and the other for *Pruned Tuple Space Search*. With the tuple pruning information, we only probe those tuples with possible matching filters.

In the previous example, we have *eight* filters in three tuples and *three* filters in the cross-product table, as shown in Table 5. To perform packet classification for a packet header, (0001,0110), the best matching prefixes of both fields are "$000*$" and "$01*$", respectively. Since the prefix, "$000*$", of f_1 is only specified in the tuple space, it indicates that its best matching prefix of the cross-product table is "$00*$". In contrast, the prefix, "$01*$", of f_2 is specified in both the cross-product table and tuple space. In the following, the combination, $\langle 00*, 01* \rangle$, is used to access the cross-product table and derive the matching filter, F_4. Moreover, the tuple pruning information of "$000*$" in f_1 shows two tuples with possible matching filters, $T_{0,4}$ and $T_{3,2}$, and that of "$01*$" in f_2 lists two tuples, $T_{3,2}$ and $T_{4,0}$. As a result, $T_{3,2}$ is probed to yield another matching filter, F_6. Assume that the filters are sorted in a descending order of priority. Then F_4 has a higher priority than F_6 and the action of F_4 is activated to process the packet. We note that it is possible for a given packet header which only matches to the filters in the cross-product table (e.g., the packet header, (0011,1101)) or tuple space (e.g., the packet header, (1111,0010)). Since each filter must be stored in either data structures, our search procedure can derive all the matching filters.

The time complexity of *Cross-producting* is O($d \log N + W^d$) and that of *Pruned Tuple Space Search* is O($d \log N$), where W is the maximum prefix length and W^d denotes the maximum number of tuples. The space complexity of *Cross-producting* is O(N^d) and that of *Pruned Tuple Space Search* is O(N). In the worst case, our scheme searches the original tuple space for the matching filters, thus the time complexity of our scheme is the same as *Pruned Tuple Space Search*. The space complexity of our scheme is affected by the value of $C_{threshold}$. Since there are at most W^d cross-product tables and each cross-product table has at most $C_{threshold}$ entries, the space complexity of our scheme is O($W^d C_{threshold}$).

3.3. Refinemens

In this section, we present three techniques to further improve the search and storage performance of our algorithm. The first technique uses multiple cross-product tables to further reduce the number of tuples and improve the search performance. This technique could enable our algorithm to support large filter databases. The second technique is adopting nested level tuple space which is a different approach to generating tuple space. The last

technique presents an adaptive approach of implementing cross-product table for storage reduction. In the following, we describe these three techniques in detail.

3.3.1. Multiple Cross-product Tables

While our algorithm could effectively reduce the number of tuples with one cross-product table, it is not enough to support a large tuple space. Therefore, it is necessary to repeatedly execute the above procedure of tuple selection in iteration. After completing a round of tuple selection, we evaluate the number of the resulted tuples to see if it is less than a threshold. If *yes*, the procedure of tuple selection is stopped. Otherwise, we repeat the procedure of tuple selection for the resulted tuples of the previous round, until the number of the resulted tuples is small enough.

Although limiting the number of tuples could achieve better search performance, it is not feasible in practice. For a given tuple space, there are usually several tuples with more filters. Inserting the filters of these tuples into the cross-product tables would lead to a significant increase in storage. Therefore, our approach to generating multiple cross-product tables merely limits the size of each cross-product table and the procedure of tuple selection stops when there is no new cross-product table. Our approach could bound the size of each generated cross-product table.

Let us reconsider the example illustrated earlier in Table 2∼4. In the first round, *three* filters, $F_3 \sim F_5$, are inserted into the cross-product table. For the rest *eight* filters, we repeat our procedure of tuple selection. In the second round, we select $T_{3,2}$ in the first iteration, and the remaining *four* filters, F_1, F_2, F_{10} and F_{11}, could be inserted into a cross-product table with *nine* entries which is less than $C_{threshold}$, 10.

To perform packet classification correctly, we must derive d best matching prefixes for each cross-product table. Therefore, each prefix must indicate the best matching sub-prefix for each cross-product table. Fortunately, we only have to store the length of the best matching sub-prefix and the value of the sub-prefix can be derived from the original matching prefix directly. Thus, the storage penalty for multiple cross-product tables is moderate and the scalability of our algorithm could be significantly improved.

3.3.2. Nested Level Tuple Space

In [26], Wang et al. improve the performance of tuple space search by categorizing the filters according to the combination of the prefix nested levels, rather than prefix lengths. The tuples which are generated based on prefix lengths are called prefix length tuples and those based on prefix nested levels are called nested level tuples. Filter categorization based on nested level tuples could effectively reduce the number of tuples since the number of distinct prefix nested levels is usually less than 4 to 6 [12, 23, 27]. With fewer tuples, the search performance could be improved as well.

To store the filters in nested level tuples, the original prefix specifications must be encoded so that distinct prefix specifications of the same nested level have the same length. Next, the filters are transformed to the encoded filters by replacing their original prefixes into the encoded prefixes and inserted into the hash table.

We can use nested level tuples to improve our algorithm. Since filter categorization based on nested level tuples does not affect the calculation of cross-producting tables, our

procedure of tuple selection is not modified. For the filters in Table 1, we can repeat the procedure of tuple selection based on nested level tuples, as shown in Table 6. In this example, three tuples are selected, $T'_{1,3}, T'_{2,2}$ and $T'_{3,1}$. Three filters, F_5, F_6 and F_7, are inserted into the cross-product table, where T' represents a nested level tuple. Although the numbers of tuples and cross-product entries are the same as the previous result based on prefix length tuples, it is usually not the case in practice. Our experimental results demonstrate that the nested level tuple selection usually has better performance, which is shown in Section 3.5.

Table 6. The Procedure of Nested Level Tuple Selection for the Filters in Table 1.

The First Iteration. ($P_1 = 8, P_2 = 7$)					
Tuple	Filters	$unique(P_1^{T'})$	$unique(P_2^{T'})$	$P_1 - unique(P_1^{T'})$	$P_2 - unique(P_2^{T'})$
$T'_{1,3}$	F_1, F_2	1	2	7	5
$T'_{2,2}$	F_3, F_4, F_8, F_9	2	2	6	5
$T'_{3,3}$	F_5	1	0	7	7
$T'_{4,2}$	F_6	1	0	7	7
$T'_{2,3}$	F_7	0	0	8	7
$T'_{3,1}$	F_{10}, F_{11}	2	1	6	6
The Second Iteration. ($P_1 = 6, P_2 = 5$)					
Tuple	Filters	$unique(P_1^{T'})$	$unique(P_2^{T'})$	$P_1 - unique(P_1^{T'})$	$P_2 - unique(P_2^{T'})$
$T'_{1,3}$	F_1, F_2	1	2	5	3
$T'_{3,3}$	F_5	1	0	5	5
$T'_{4,2}$	F_6	1	0	5	5
$T'_{2,3}$	F_7	0	0	5	5
$T'_{3,1}$	F_{10}, F_{11}	2	1	4	4
The Third Iteration. ($P_1 = 5, P_2 = 3$)					
Tuple	Filters	$unique(P_1^{T'})$	$unique(P_2^{T'})$	$P_1 - unique(P_1^{T'})$	$P_2 - unique(P_2^{T'})$
$T'_{3,3}$	F_5	1	0	4	3
$T'_{4,2}$	F_6	1	0	4	2
$T'_{2,3}$	F_7	0	0	4	3
$T'_{3,1}$	F_{10}, F_{11}	2	1	3	2

Although nested level tuple selection has better performance, adopting nested level tuple would incur an extra problem in supporting incremental updates. Unlike prefix length tuples, the combination of prefix nested levels of each filter correlates to the other filters. For the case of inserting a new filter with a new prefix specification or deleting an existing filter with at least one unique prefix specification, the prefix nested level of the existing prefixes might be changed. In the worst case, the prefix nested levels of all the filters are modified and the tuple space must be reconstructed. We address the issue of supporting incremental updates in Section 3.4..

3.3.3. Adaptive Implementation of Cross-product Table

As mentioned above, the cross-product table can be implemented in a direct access array or a hash table. For the example in Table 6, there are three filters, F_5, F_6 and F_7, inserted into the cross-product table. The array-based cross-product table requires *six* entries, $\langle 0*, 1* \rangle$,

$\langle 0*, 01* \rangle$, $\langle 0*, 11* \rangle$, $\langle 00*, 1* \rangle$, $\langle 00*, 01* \rangle$ and $\langle 00*, 11* \rangle$. The alternate requires only *four* entries, $\langle 00*, 11* \rangle$, $\langle 000*, 11* \rangle$, $\langle 000*, 01* \rangle$ and $\langle 111*, 11* \rangle$. The entries, $\langle 00*, 01* \rangle$ and $\langle 111*, 01* \rangle$, are removed since both entries do not match any filters.

While the hash-based cross-product table usually requires fewer entries, it might incur hash collisions and result in performance degradation. To alleviate the problem of hash collisions, a hash table usually requires *two* or more times the entries of stored entries. Moreover, the storage reduction may not be in effect due to the correlation between prefixes. Consider another *three* filters, F_3, F_4 and F_5, which are inserted into cross-product table in Table 4, the numbers of entries in either the array-based or hash-based cross-product table are the same. Therefore, we choose the one with smaller storage requirement to minimize the memory consumption. Since the number of entries for the array-based cross-product table can be estimated by multiplying the numbers of field specifications, we always generate the hash-based cross-product table to decide which data structure is used.

3.4. Incremental Updates

A filter update can either be an insertion or a deletion. For our scheme, both types of updates would affect one of the data structures (tuple space or cross-product tables). These data structures are also related to the data structure of the best-matching prefix search. Since the update procedure on the best-matching prefix data structure has been studied extensively [28], we focus on the update procedure of the proposed data structures. Our algorithm uses two different data structures, *tuple space* and *cross-product table*. In the following, we present the procedures for each type of data structures and combine these procedures in the end.

We consider updating the data structure of tuple space first. As mentioned above, the tuple space can be implemented with prefix length tuples or nested level tuples. Since prefix length tuples have been demonstrated that filter updates can be accomplished within 100μsec [29], we focus on updating nested level tuples. For a filter insertion with at least one new prefix specification, the prefix nested levels of those prefixes which are successive to the new prefix are incremented by *one*. Thus, the nested level combinations of those filters which specify the prefixes with the updated nested levels must be modified as well. In the worst case, we have to update the nested level combinations of all the filters. To avoid the reconstruction of tuple space, we can build another tuple space for those filters with new prefix specifications. For the example in Table 6, we have three nested level tuples, $T'_{1,3}, T'_{2,2}$ and $T'_{3,1}$, in a tuple space. If a new filter, $\langle 11*, 01* \rangle$, is inserted into the tuple space, the new filter would result in the nested level of prefixes, "$111*$" and "$1111*$", incremented by *one*. Consequently, the nested level combinations of F_9 and F_{11} are changed to $T'_{3,2}$ and $T'_{4,1}$, respectively, and the new filter is inserted into $T'_{2,2}$.

To avoid the problem of changing the nested levels of the existing filters, we generate a new tuple space for those filters with a new prefix specification. The new filter with at least one new prefix specification is inserted into the new tuple space which means the original tuple space remains unchanged. The new tuple space also can be implemented with prefix length tuples. Since the tuple space with prefix length tuples supports incremental updates, there are at most two tuple spaces, one based on nested level tuples and the other based on prefix length tuples. In addition, we keep the prefix specification that is only specified by a

deleted filter. Thus, filter deletion would not change nested levels of the existing prefixes.

Next, we consider the problem of updating the cross-product table. Again, the cross-product table can be implemented as a direct access array or a hash table. The implementation based on direct access array can increase the access rate, but it is usually not updatable due to its consecutively allocated entries. Although the cross-product table based on direct access array cannot support filter insertion, it can support filter deletion by *keeping the deleted filter and disabling its action*. As a result, the subsequent processing of the incoming packet would not be affected by the deleted filter. The deleted filter will be removed after reconstructing the cross-product table. For the hash-based cross-product table, we use the same approach to handle filter deletion. Although it is feasible to support filter insertion in the hash-based cross-product table, the cross-product table still has to include $(N+1)^d - N^d$ extra cross-product entries in the worst case. Therefore, we directly insert new filters into tuple space to eliminate the computation cost of generating cross-product entries.

After introducing the update procedures for each data structure, now let us combine all the update procedures. For a filter deletion, we decide in which data structure the deleted filter is stored. If the filter is stored in the tuple space, we delete it directly but keep its prefix specifications to maintain the original tuples. Otherwise, we use the approach of disabling filter action to update the corresponding cross-product table. For a filter insertion, we only store the new filter in the tuple space. If the tuple space consists of prefix length tuples, we can insert the new filter directly. However, for the nested level tuples, a new filter whose new prefix specification affects the original prefix nested level is inserted in a new tuple space based on prefix length tuples to avoid updating an overwhelming number of tuples. Since both filter insertion and deletion of our scheme updates prefix length tuples, the update performance of our scheme is comparable to that of *Pruned Tuple Space Search*.

3.5. Performance Evaluation

In this section, we describe how our algorithm performs on both real and synthetic databases in terms of its speed and storage. We use three five-dimensional real databases in [30]. Since the largest real database contains only 1,550 filters, we also include five-dimensional synthetic databases, which are generated by a publicly available tool, *ClassBench* [31], to test the scalability of our scheme. Each synthetic database has different characteristics that are extracted from one of the twelve real databases [31]. Therefore, we can investigate the performance of packet classification under different circumstances or with different devices, such as ISP peering routers, backbone core routers, enterprise edge routers and firewalls [31]. We also use *ClassBench* to generate the tested packet trace for the designated filter databases.

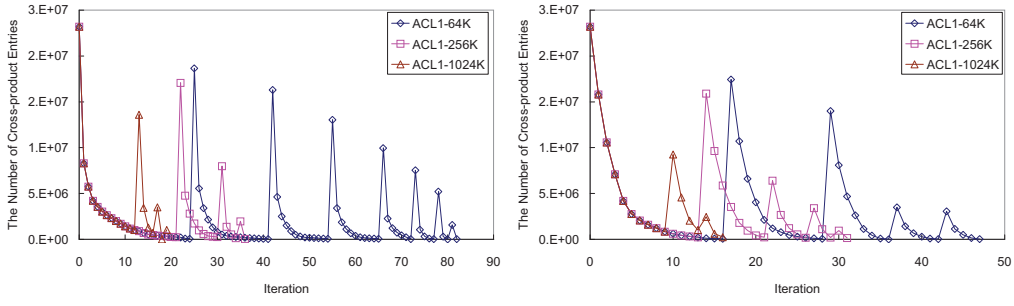
The performance evaluation has four parts. The first part evaluates the effectiveness of our scheme by presenting the trade-off between the number of tuples and the sizes of cross-product tables in each iteration of tuple selection. Next, we show the performance of our scheme with different refinements. The third part is a performance study that compares the numerical results of the proposed scheme with other existing schemes. While the first three evaluations are based on real filter databases, the last evaluation is carried out upon the synthetic filter databases. In the last two experiments, the cross-product table of our

algorithm can be stored in either a direct access array or a hash table, depending on the sizes of data structures. We also provide the performance of our algorithm using prefix length tuples (PLTs) and nested level tuples (NLTs) for a better illustration of their differences. Since our scheme must perform one-dimensional searches, we adopt multiway search tree [32] for one-dimensional search in our experiments. Our main reason for choosing this data structure is its scalability with respect to the number of prefixes and prefix lengths.

3.5.1. Trade-off Between the Number of Tuples and the Sizes of Cross-Product Tables

We start the performance evaluation by presenting the trade-off between the number of tuples and the sizes of cross-product tables. We did this evaluation on three real filter databases that are publicly available in [30]. For each database, we repeat the procedure of tuple selection until we cannot create any cross-product tables. Therefore, we would generate multiple cross-product tables. For each filter database, we also show the influence of cross-product table size by adjusting the value of $C_{threshold}$. We use three different values of $C_{threshold}$, 64K, 256K and 1024K.

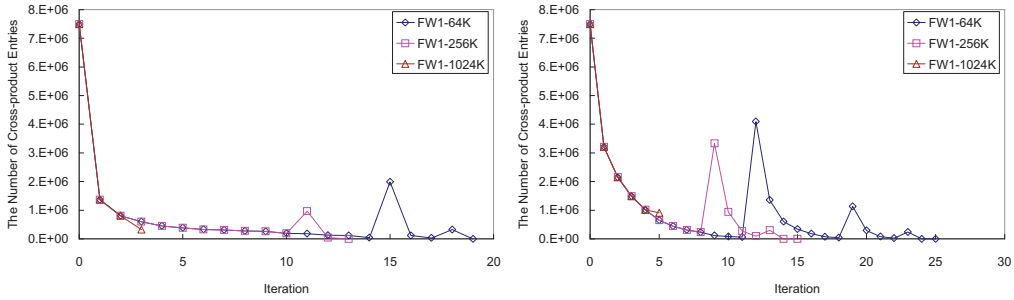
As shown Fig. 1$\sim$3, the number of cross-product entries varies from several millions to about half billion initially. With our algorithm, the number of cross-product entries is decreased by removing the filters of selected tuples from the cross-product table. The procedure of tuple selection usually reduces the most cross-product entries in the first few iterations. After generating a cross-product table whose size is less than $C_{threshold}$, we start another round of tuple selection by moving all filters from tuple space to a new cross-product table. Therefore, the number of cross-product entries raises again, and the procedure of tuple selection repeats until cross-product tables with less than $C_{threshold}$ entries are generated. Our experimental results show that the number of cross-product tables usually ties to the number of filters. Since IPC1 has the most filters, it also requires more iterations of tuple selection and results in more cross-product tables than the other databases. In addition, our algorithm based on NLTs requires much less iterations than that based on PLTs for ACL1 and IPC1, but slightly more than that based on PLTs for FW1. Therefore, using NLTs in our algorithm could shorten the time for constructing our data structures.



(a) Prefix Length Tuples

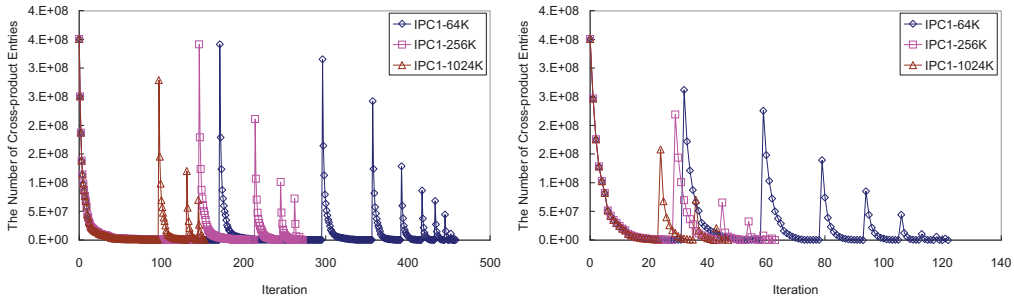
(b) Nested Level Tuples

Figure 1. The size of cross-product table in each iteration of tuple selection for ACL1.



(a) Prefix Length Tuples (b) Nested Level Tuples

Figure 2. The size of cross-product table in each iteration of tuple selection for FW1.



(a) Prefix Length Tuples (b) Nested Level Tuples

Figure 3. The size of cross-product table in each iteration of tuple selection for IPC1.

3.5.2. Comparative Analysis of Our Refinements

In this subsection, we show the performance of our scheme with the first and third refinements presented in Section 3.3.. We fix the value of $C_{threshold}$ to 64K based on previous evaluation. Moreover, we only use the array-based cross-product tables. To test the refinement of generating multiple cross-product tables, we use two different configurations to evaluate our scheme. The first configuration is a basic scheme which only generates one cross-product table, and the other uses the refinement to generate multiple cross-product tables. We show the numerical results of our scheme with both configurations in Table 7 $\sim$9. Table 7 and 8 show the storage performance of two configurations for PLT and NLT, respectively. With the refinement of generating multiple cross-product tables, the number of tuples could be significantly reduced, regardless PLTs or NLTs. Although the required storage is increased since cross-product tables usually store redundant entries, the search performance of our scheme is improved with less data structures, as shown in Table 9. Therefore, there exists a tradeoff between storage performance and the number of generated cross-product tables.

Next, we consider the influence of cross-product table implementations on the storage

Table 7. The Storage Performance of Our Scheme with One or More Cross-product Tables. (PLT)

Real Database	Original Filters	Original Tuples	One Cross-product Table			Multiple Cross-product Tables			
			Filters in PLTs	PLTs	Storage	CP Tables	Filters in PLTs	PLTs	Storage
ACL1	752	75	1,722	25	353.73	7	448	2	537.71
FW1	269	206	241	28	320.75	2	163	2	338.59
IPC1	1,550	219	1,528	192	351.69	10	275	1	692.85

Table 8. The Storage Performance of Our Scheme with One or More Cross-product Tables. (NLT)

Real Database	Original Filters	Original Tuples	One Cross-product Table			Multiple Cross-product Tables			
			Filters in NLTs	NLTs	Storage	CP Tables	Filters in NLTs	NLTs	Storage
ACL1	752	31	1,730	11	351.29	4	1,060	5	413.00
FW1	269	32	805	20	327.03	3	141	2	399.70
IPC1	1,550	83	2,072	60	313.70	8	398	2	587.63

Table 9. The Speed Performance of Our Scheme with One or More Cross-product Tables.

Real Database	One Cross-product Table				Multiple Cross-product Tables			
	PLT		NLT		PLT		NLT	
	AMA	WMA	AMA	WMA	AMA	WMA	AMA	WMA
ACL1	16.39	22	16.68	22	15.75	19	15.98	21
FW1	14.59	19	14.02	19	13.79	17	14.96	19
IPC1	18.12	30	18.40	28	18.3	24	17.9	24

performance of our scheme for the case of generating multiple cross-product tables. As listed in Table 10, the array-based cross-product tables lead to less storage requirements than hash-based cross-product tables. Since the entry stored in hash tables must keep the original content for comparison, the array-based cross-product tables have better storage efficiency. However, the hash-based cross-product table might be preferable to array-based cross-product table for storing a large amount of filters since the number of redundant entries in the latter would increase. By combining both implementations in our scheme, we could achieve the best storage requirements as compared to a single implementation of cross-product tables.

Table 10. The Storage Performance of our Scheme with Different Implementations of Cross-product Tables.

Real Database	PLT			NLT		
	Hash	Array	Hybrid	Hash	Array	Hybrid
ACL1	1,156.98	537.71	422.42	847.42	413.00	377.55
FW1	647.78	338.59	336.97	602.58	399.70	396.39
IPC1	1,542.67	692.85	605.39	693.92	587.63	495.73

3.5.3. Comparative Analysis of Our Scheme and Previous Schemes

We did the third evaluation by comparing our scheme with other notable schemes. The schemes from previous work include *ABV* [12], *HyperCuts* [10], *RFC* [2] and *Pruned Tuple Space Search* (*PTSS*) [8]. For the *ABV* scheme, the aggregate size is 32 bits, and the memory width is 256 bits. *Hypercuts* adopts the setting in which the space factor is 1 and bucket size is 32. For our scheme, the value of $C_{threshold}$ is set to 64K, and all of the refinements are used. The size of the hash table is set to two times the number of the stored entries, and the hash collision is resolved by linked-list chaining. In the following results, the performance metrics include the required storage in kilobytes and the numbers of memory accesses in the average case (AMA) and the worst case (WMA). The required storage includes all necessary data structures for performing packet classification; hence, the required space of d multi-way search trees is also included. The values of AMA are derived by dividing the total number of memory accesses for classifying every packet header in the trace from the number of classified packet headers.

Table 11 shows the storage performance of the proposed scheme and other existing schemes, and Table 12 lists the speed performance. The experimental results show that our algorithm has better storage performance than *RFC*, but worse than the other schemes. On the contrary, our algorithm has better search performance than most schemes, except *RFC*. The results show the trade-off between storage and speed performance lying in the algorithms of packet classification. Although our algorithm does not show significant advantages over the existing algorithms, most of the existing schemes, except for *PTSS*, cannot support incremental updates [4]. Therefore, our algorithm has better feasibility than the existing schemes.

Table 11. Storage Performance of the Existing Algorithms Using Real Databases.

Real Database	Original Filters	ABV	HyperCuts	RFC	PTSS	Our Scheme	
						PLT	NLT
ACL1	752	296.34	29.45	497.62	293.17	422.42	377.55
FW1	269	263.73	33.41	1,094.55	274.39	336.97	396.39
IPC1	1,550	331.46	142.18	8,984.18	299.15	605.39	495.73

Table 12. Speed Performance of the Existing Algorithms Using Real Databases.

Real Database	ABV		HyperCuts		RFC		PTSS		Our Scheme			
									PLT		NLT	
	AMA	WMA	AMA	WMA	AMA	WMA	AMA	WMA	AMA	WMA	AMA	WMA
ACL1	35.13	44	14.79	22	11	11	20.18	32	15.75	19	15.98	21
FW1	20.49	30	21.37	46	11	11	85.25	124	13.79	17	14.96	19
IPC1	26.14	50	22.44	49	11	11	25.72	48	18.30	24	17.90	24

We also note that the performance difference between using PLTs or NLTs in our algorithm is usually less than 10%. Since tuple space based on PLTs could provide better renewability than based on NLTs, we can use the PLT-based tuple space for network devices with frequent updates, such as QoS routers. On the other hand, NLT-based tuple space requires less tuples than PLT-based tuple space; therefore, it could reduce the overhead of hash table management. Therefore, the NLT-based tuple space is suitable for rather static network devices, such as firewalls or network intrusion detection systems.

3.5.4. Scalability Evaluation

To test the scalability of our algorithm, we use twelve synthetic databases for further evaluation. The synthetic databases are generated by *ClassBench* with default settings. Each database is initialized with 16,000 filters. However, some filters might be redundant, and, as a result, the actual number of filters in each database is usually less than 16,000. In this experiment, we set the value of $C_{threshold}$ to 1024K for handling large databases.

We show the storage and speed performance in Table 13 and 14, respectively. The experimental results of *RFC* are not listed since it takes up too much storage. As shown in Table 13, the trade-off between storage and speed performance remains in the experimental results. Although our algorithm still requires more storage than *PTSS*, its storage performance outperforms that of *ABV* and *HyperCuts* in most cases. Moreover, our scheme provides the best search performance among these schemes. Therefore, by inheriting the characteristics of *Cross-producting*, our algorithm could achieve superior search performance and better scalability.

Table 13. Storage Performance of the Existing Algorithms Using Synthetic Databases.

Synthetic Database	Original Filters	ABV	HyperCuts	PTSS	Our Scheme	
					PLT	NLT
ACL1	15,926	36,523.81	369.18	1,010.32	1,563.63	1,798.65
ACL2	15,447	55,141.81	1,359.68	2,055.93	2,023.45	2,684.40
ACL3	14,729	7,803.02	2,721.50	726.90	3,294.19	3,288.41
ACL4	15,405	13,306.42	1,985.26	793.31	4,996.05	4,046.99
ACL5	10,379	4,545.09	294.45	534.08	635.45	557.82
FW1	14,898	32,494.13	23,883.84	2,205.53	1,932.36	4,427.26
FW2	15,501	39,885.48	10,859.94	1,962.20	1,673.43	2,619.59
FW3	14,297	27,532.10	27,172.06	1,878.96	1,715.81	2,023.88
FW4	13,856	31,231.93	10,717.61	2,855.36	5,096.11	5,227.80
FW5	14,009	26,260.04	19,503.95	1,750.74	1,681.79	2,307.11
IPC1	14,954	16,116.21	3,557.90	716.50	3,260.24	3,199.90
IPC2	16,000	44,405.32	12,450.29	1,893.48	1,076.68	1,076.58

Table 14. Speed Performance of the Existing Algorithms Using Synthetic Databases.

Synthetic Database	ABV		HyperCuts		PTSS		Our Scheme			
							PLT		NLT	
	AMA	WMA	AMA	WMA	AMA	WMA	AMA	WMA	AMA	WMA
ACL1	33.30	44	21.07	56	16.49	32	14.13	18	14.17	17
ACL2	36.55	50	21.72	132	28.49	57	10.37	13	11.6	15
ACL3	52.55	84	22.78	119	41.87	82	19.31	36	19.47	32
ACL4	50.00	84	21.78	92	39.03	77	19.58	35	22.85	40
ACL5	33.72	57	28.28	60	19.18	36	15.28	23	15.22	22
FW1	47.33	56	22.52	221	77.96	113	10.96	13	11.07	13
FW2	33.28	34	21.12	43	11.15	15	7.73	10	7.73	10
FW3	51.71	61	23.47	201	77.28	112	9.97	12	10.09	12
FW4	53.47	87	24.22	639	101.31	187	13.93	16	13.03	14
FW5	51.65	61	23.95	182	106.61	142	9.86	12	9.84	12
IPC1	44.05	65	22.44	87	21.52	44	20.15	33	18.92	30
IPC2	32.34	33	17.14	41	8.36	9	6.08	8	6.08	8

4. Hashing and Bit Vectors

4.1. Algorithm

In this section, we adopt the BV algorithm to improve the search performance of *Pruned Tuple Space Search*. Since the filters in the same tuple can be accessed with one hash probe, tuple space search is suitable for a large amount of filters with orderly length combinations, which results in fewer tuples. In contrast, the *BV* algorithm could handle the filters with disordered length combinations. However, the *BV* algorithm is limited in the number of filters. Therefore, we divide the filters into two categories, where one category is searched by *BV* and the other is searched by tuple space search. We further propose an algorithm to minimize the number of tuples for a predetermined memory space of bit vectors.

We use the filters in Table 1 as an example to further illustrate our idea. Since there are 15(=8+7) bit vectors and each bit vector has 11 bits, the total storage requirement of bit vectors is 165 bits. The 11 filters can be distinguished into six prefix-length combinations and each combination corresponds to one hash table. Assume that we select the filters of length combination $\langle 3, 2\rangle$, $R_6 \sim R_9$, for tuple space search. The remaining seven filters have 11 distinct prefix specifications; hence, the required bit vector storage is reduced to 77 bits. If we choose further the filters of another length combination $\langle 4, 0\rangle$, R_{10} and R_{11}, then the bit vector storage would be reduced to 40 bits. While there are only two tuples in the resulted tuple space, the bit vector storage is reduced to less than a quarter. Since the search cost of two hash tables is low, the search and storage performance could be improved simultaneously.

We describe the filter categorization procedure of our scheme in the following. First, we define a threshold value for the bit vector storage, $S_{threshold}$. The threshold value can be set to the size of on-chip static random access memory (SRAM) to improve the access performance. Next, we derive the original storage requirement of bit vectors by multiplying the numbers of filters by the number of distinct prefix specifications in the filter database. We denote the number of filters as N and the number of distinct prefixes as P. Then, the storage requirement of bit vectors, S, can be expressed as $S = N \times P$. In the third step, we construct a tuple space according to the length combinations of filters. Assume that the number of the filters in tuple T_i is denoted as N_{T_i} and the number of the distinct field specifications is denoted as P_{T_i}. The storage requirement for the filters in T_i is denoted as S_{T_i}. Then, the maximum value of S_{T_i} is equal to $N_{T_i} \times P$ when the P_{T_i} prefixes are also specified by the filters of other tuples. Also, the maximum value of S_{T_i} is increased to $S_{T_i} = N \times P - (N - N_{T_i}) \times (P - P_{T_i})$ when none of the prefixes in T_i is specified by the filters of other tuples. Therefore, we calculate the numbers of the filters and unique prefix specifications for each tuple. The number of unique prefix specifications, P'_{T_i}, indicates those which are *only* specified in T_i. The value of S_{T_i} is thus expressed as $S_{T_i} = N \times P - (N - N_{T_i}) \times (P - P'_{T_i})$. In the fourth step, we extract the filters of the tuple T_i with the largest S_{T_i} value for tuple space search. Such selection can maximize storage reduction for the bit vectors while only one extra hash table is generated for tuple space search. After deducting the required bit vector storage of the selected tuple, we check whether the new storage requirement, $(N - N_{T_i}) \times (P - P'_{T_i})$, is smaller than $S_{threshold}$. If the answer is *yes*, the procedure of filter categorization is accomplished. Otherwise, the values of N and

P are updated and the fourth and fifth steps are repeated until the storage threshold can be satisfied.

We use the filters in Table 1 as an example to illustrate our procedure of filter categorization. We assume that $S_{threshold}$ equals 16. Table 15 lists three iterations that calculate the resulted storage of bit vectors. Initially, the value of N is 11 and that of P is 15. Therefore, S is equal to 165 bits. In the first iteration, selecting the filters from $T_{3,2}$ would result in the minimum storage requirement. Therefore, the values of N and P are updated to 7 and 11, respectively. Since the filters of $T_{3,2}$ have been selected in this iteration, they are not considered in the subsequent iterations. In the second iteration, both tuples, $T_{0,4}$ and $T_{4,0}$, result in the same storage requirement. In this case, we randomly choose $T_{4,0}$ and update the values of N and P. The procedure stops after the third iteration which results in a storage requirement of 15 bits by selecting $T_{0,4}$. As a result, there are three tuples with eight filters in the resulted tuple space. The remaining three filters are stored in the form of bit vectors.

Table 15. Three Iterations of Our Filter Categorization Procedure for the Filters in Table 1.

Tuple	Filters	Initiation		First Iteration		Second Iteration		Third Iteration	
		N_{T_i}	P'_{T_i}	$N - N_{T_i}$	$P - P'_{T_i}$	$N - N_{T_i}$	$P - P'_{T_i}$	$N - N_{T_i}$	$P - P'_{T_i}$
$T_{0,4}$	R_1, R_2	2	3	9	12	5	8	**3**	**5**
$T_{1,1}$	R_3	1	1	10	14	6	10	4	7
$T_{1,2}$	R_4	1	0	10	15	6	11	4	8
$T_{2,2}$	R_5	1	1	10	14	6	10	4	7
$T_{3,2}$	$R_6 \sim R_9$	4	4	**7**	**11**	**NA**	**NA**	**NA**	**NA**
$T_{4,0}$	R_{10}, R_{11}	2	3	9	12	**5**	**8**	**NA**	**NA**

We note that the calculation of S_{T_i} might underestimate the required storage of T_i. While the filters of several tuples are selected for tuple space search, the prefix specifications of the remaining filters which are originally not unique might become unique. Therefore, in practice, the storage performance might be better than the estimated one. In the previous example, the calculation for the tuple $T_{1,2}$ in the third iteration is inexact since the value of $P - P'_{T_i}$ should be *seven*, rather than *eight* in Table 15. Although an accurate calculation could be achieved, we believe that such overestimation of storage requirement may not significantly affect the result of filter categorization. The main reason is due to the uneven filter distribution in the existing filter databases. Most filters are distributed among few tuples. Since these tuples are likely to be selected in the first few iterations, the calculation for the remaining tuples is less important. Moreover, the extra computation for a more precise calculation is costly and may not be feasible in practice.

To initiate the procedure of filter categorization, we construct d prefix tries and derive necessary values by traversing these tries. In each following iteration, we simply compare the cost of each tuple. Therefore, the time complexity of filter categorization is O(dNW). After finishing the procedure of filter categorization, each filter must be stored in one of the data structures, bit vectors or hash tables. Their storage locations are not changed unless both data structures are reconstructed.

Next, we construct searchable data structures for the two categories of filters. For the filters stored in bit vectors, d one-dimensional data structures of the best matching prefixes (BMPs) are built. Each prefix node is associated with a bit vector whose length is equal to the number of filters in the category. We also insert an aggregate bit vector for each bit vector to further improve its speed [12]. The filters of the other category are inserted into a tuple space. We use *PTSS* [8] to search these filters since it also uses BMP searches to decide which tuples are probed. We can combine the BMP searches of both categories to facilitate faster search performance.

The search procedure acts as follows. First, d one-dimensional searches are carried out to determine the BMP of each field. Subsequently, the corresponding aggregated bit vectors are accessed to decide which words of the matching bit vectors are fetched. The tuple pruning information is also extracted from the BMP, and those tuples which appear in d BMPs are probed for a possible filter match. The tuple pruning information can be represented in the form of bit vectors. Combining the results of both searches can derive all the matching filters.

4.2. Incremental Updates

A filter update can be either a deletion or an insertion. For our scheme, both types of updates would affect one of the data structures, (*PTSS* or *ABV*). Both data structures also include d BMP data structures. Since the update procedures for various BMP data structures have been studied extensively, we put our emphasis on the update procedures of the multidimensional data structures. There are two types of updates in our scheme, one for tuple space and the other for bit vectors. Since the update procedures for tuple space have been presented in the previous section, we describe the update procedures of bit vectors in the following.

1. *Delete Filter from BV*: Deleting a filter from *BV* is more difficult than from tuple space. Due to the property that a value matching to a prefix, p, also matches to p's subprefixes, the corresponding bit vector of each prefix must include the same set bits as those of its subprefixes [12]. Therefore, updating the bit vector of prefix, "$*$", will affect all the bit vectors of the same field and every bit in bit vectors are updated in the worst case. To alleviate the update cost, we merely set the bit corresponding to the deleted filter to *zero*. Therefore, at most P_{BV} words are updated for each filter deletion, where P_{BV} denotes the number of distinct prefixes for the filters stored in bit vectors. Since our scheme of filter categorization has reduced the value of P_{BV} to minimize the storage of bit vectors, the update cost is reduced as well. We note that our approach would degrade the storage efficiency; however, the storage efficiency can be restored by combining the procedure of filter insertion listed below.

2. *Insert Filter into BV*: In [12], the authors have reported that inserting a filter into bit vectors is slow in the worst case, but fast on the average. The reason for the slow updates comes from inserting a new filter which has at least one *wildcard* field since all bit vectors of these *wildcard* fields must be updated. In addition, inserting filters with new prefix specifications also requires the generation of d new bit vectors. Hence, the number of maximum updated words is $(P_{BV} + dN_{BV}/B)$, where N_{BV} denotes the number of filters stored in *BV* and B represents the memory width. To

avoid the worst-case update performance of filter insertion, the new filters with new prefix specifications are inserted into a tuple space. The other new filters without new prefix specifications are inserted into *BV*. We reuse the bits of previously deleted filters to restore the storage efficiency. Since no new bit vectors are generated, the number of updated words remains P_{BV}.

In sum, our update procedures perform filter deletion directly while adaptively select the data structures for filter insertion. By combining *BV* and *PTSS*, we can avoid the high cost of filter insertion/deletion in the worst case. Therefore, our update procedures have superior flexibility because of the unique design of hybrid data structures. Although our approach would degrade the storage efficiency of *BV* in the case of a heavy load of filter deletions, it would be reasonable to reconstruct bit vectors with a storage efficiency lower than a certain threshold due to the infrequency of filter updates [12, 27].

4.3. Performance Evaluation

In this section we describe how our scheme performs on different filter databases in terms of speed and storage. The performance evaluation is divided into two stages. The first stage evaluates the effectiveness of our scheme by presenting the tradeoff between the number of tuples and BV storage requirement. The second is a performance study that compares the numerical results of our scheme with several notable algorithms.

4.3.1. Trade-off Between the Number of Tuples and BV Storage Requirement

In this section, we show the tradeoff between the number of tuples and BV storage to illustrate the effectiveness of our scheme. For three real filter databases, Figure 4 depicts that the required BV storage is reduced by increasing the number of tuples, where the BV storage is measured in kilobytes. When all the filters are stored in tuple space, the BV storage is reduced to *zero*. The figure also indicates that for those filter databases with a large amount of tuples, our scheme is useful since we can restrict most filters to few tuples and only few filters are stored in bit vectors. Therefore, the search performance of tuple space search and storage performance of bit vectors could be improved simultaneously.

We further list the required BV storage for different number of tuples in Table 16 for real databases. The term, $|TS|$, denotes the number of tuples in tuple space. While $|TS = 0|$, our scheme acts as *ABV*; hence, our scheme has the same BV storage as *ABV*. By removing the filters of the selected tuple from bit vectors in each iteration, the BV storage is reduced as well. Our results show that we can achieve a significant reduction of BV storage by removing filters in the first few tuples, which conforms to the observation that the filters in tuple space are unevenly distributed [33]. Since our scheme of filter categorization prefers those in the tuple with a large amount of filters and distinct prefixes, we can greatly reduce the BV storage with only a few tuples to achieve better storage efficiency.

Next, we use twelve synthetic databases for further evaluation. Each database is initialized with 16,000 filters. After removing the redundant filters, the actual number of filters in each database is usually less than 16,000. Unlike real databases, the synthetic databases require much large storage for bit vectors. Therefore, we constrain the required BV storage to

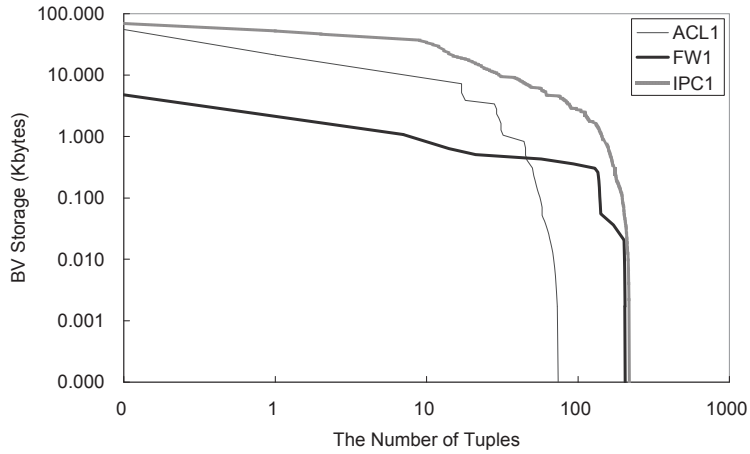


Figure 4. The Tradeoff between the Number of Tuples and BV Storage Requirement for Real Filter Databases.

Table 16. Tradeoff between the Number of Tuples and BV Storage for Real Databases.

Databases	Original Filters	Original Tuples	BV Storage for Different Number of Tuples					
			$\|TS\| = 0$	$\|TS\| = 1$	$\|TS\| = 2$	$\|TS\| = 4$	$\|TS\| = 8$	$\|TS\| = 16$
ACL1	752	79	55.15	21.17	7.20	3.87	1.99	0.31
FW1	269	221	4.76	1.07	0.63	0.43	0.23	0.01
IPC1	1,550	244	69.73	52.49	46.68	34.07	22.67	14.89

evaluate the number of tuples. As shown in Table 17, the experimental results show that the BV storage could be reduced to less than 1,024 kilobytes with only 15% tuples, except for the case of IPC2 which only has *eight* tuples. Moreover, we only need 30% of the original tuples to obtain a BV storage of 128 kilobytes. Accordingly, a small amount of BV storage is enough to eliminate most tuples in the original tuple space.

4.3.2. Comparative Analysis

Next, we compare our scheme with several notable algorithms. The algorithms include *ABV* [12], *HyperCuts* [10], *PTSS* [8] and *RFC* [2]. For *ABV* and our scheme, the aggregate size is 32 bits, and the memory width is 256 bits. *Hypercuts* adopts the setting in which the space factor is 1 and bucket size is 32. Based on our evaluation to the tradeoff between the BV storage and the number of tuples, we limit the size of bit vectors to 1,024 kilobytes in the following experiments. However, for the database whose original BV storage is less than the threshold, we still extract *four* tuples to improve storage efficiency.

The evaluation begins with the real databases. The storage and speed performance of our scheme and other existing algorithms are shown in Table 18 and Table 19, respectively. As compared to *ABV*, our scheme yields a better storage and speed performance, except for FW1. This is because there are only few filters in FW1. Therefore, the storage reduction of bit vectors is offset by the increased storage for the hash tables. For *PTSS*, the new scheme

Table 17. Tradeoff between the Number of Tuples and BV Storage for Synthetic Databases.

Databases	Original Filters	Original Tuples	Original BV Storage	The Number of Tuples for Different BV Storage				
				128 KB	256 KB	512 KB	1,024 KB	2,048 KB
ACL1	15,926	109	35,044.32	25	20	16	12	8
ACL2	15,447	229	43,531.43	46	38	31	26	21
ACL3	14,729	545	7,201.66	112	83	56	34	16
ACL4	15,405	611	12,447.35	133	102	72	45	25
ACL5	10,379	74	4,140.45	15	9	5	2	1
FW1	14,898	237	23,225.51	11	9	7	5	2
FW2	15,501	34	22,845.59	10	8	7	5	4
FW3	14,297	168	21,191.83	3	2	2	2	1
FW4	13,856	436	26,533.64	16	12	10	8	5
FW5	14,009	211	16,584.86	7	5	3	2	2
IPC1	14,954	362	14,527.57	103	87	68	48	30
IPC2	16,000	8	8,119.70	4	3	3	2	2

has better search performance since the number of tuples has been reduced significantly. As compared to other algorithms, our scheme needs more storage than *HyperCuts* but less than *RFC*. However, the search performance of our scheme is only worse than that of *RFC*. The results show the trade-off between storage and speed performance in the algorithms of packet classification. In general, the algorithms consuming more memory space would have better search performance. Nevertheless, the feasibility of these algorithms would be a major hurdle for supporting large filter databases.

Table 18. Storage Performance of the Existing Algorithms Using Real Databases.

Databases	Filters	ABV	HyperCuts	PTSS	RFC	Our Scheme
ACL1	752	296.34	29.45	293.17	497.62	291.94
FW1	269	263.73	33.41	274.39	1,094.55	266.48
IPC1	1,550	331.46	142.18	299.15	8,984.18	319.69

Table 19. Speed Performance of the Existing Algorithms Using Real Databases.

Databases	ABV		HyperCuts		PTSS		RFC		Our Scheme	
	AMA	WMA	AMA	WMA	AMA	WMA	AMA	WMA	AMA	WMA
ACL1	35.13	44	14.79	22	20.18	32	11	11	17.52	22
FW1	20.49	30	21.37	46	85.25	124	11	11	13.52	20
IPC1	26.14	50	22.44	49	25.72	48	11	11	22.39	35

For synthetic databases, we show the storage and speed performance in Table 20 and 21, respectively. The experimental results of *RFC* are not listed since it takes up too much storage. As shown in Table 20, the trade-off between storage and speed performance remains in the experimental results, but our scheme features relatively stable storage requirements. While the required storage of *ABV* and *HyperCuts* varies for different databases, our scheme could retain the storage requirement in the same level. In addition, our scheme yields a better search performance than the existing algorithms, as shown in Table 21. Therefore, our

scheme has better feasibility than the existing algorithms in practice.

Table 20. Storage Performance of the Existing Algorithms Using Synthetic Databases.

Databases	Filters	ABV	HyperCuts	PTSS	Our Scheme
ACL1	15,926	36,523.81	369.18	1,010.32	2,331.59
ACL2	15,447	55,141.81	1,359.68	2,055.93	2,984.88
ACL3	14,729	7,803.02	2,721.50	726.90	1,888.43
ACL4	15,405	13,306.42	1,985.26	793.31	1,981.69
ACL5	10,379	4,545.09	294.45	534.08	1,611.58
FW1	14,898	32,494.13	23,883.84	2,205.53	2,564.19
FW2	15,501	39,885.48	10,859.94	1,962.20	2,654.18
FW3	14,297	27,532.10	27,172.06	1,878.96	1,851.56
FW4	13,856	31,231.93	10,717.61	2,855.36	2,653.86
FW5	14,009	26,260.04	19,503.95	1,750.74	2,483.67
IPC1	14,954	16,116.21	3,557.90	716.50	2,018.23
IPC2	16,000	44,405.32	12,450.29	1,893.48	1,838.34

Table 21. Speed Performance of the Existing Algorithms Using Synthetic Databases.

	ABV		HyperCuts		PTSS		Our Scheme	
Databases	AMA	WMA	AMA	WMA	AMA	WMA	AMA	WMA
ACL1	33.30	44	21.07	56	16.93	32	15.06	25
ACL2	36.55	50	21.72	132	28.49	57	11.09	20
ACL3	52.55	84	22.78	119	41.87	82	23.10	43
ACL4	50.00	84	21.78	92	39.03	77	24.03	43
ACL5	33.72	57	28.28	60	19.18	36	19.12	40
FW1	47.33	56	22.52	221	77.96	113	21.00	26
FW2	33.28	34	21.12	43	11.15	15	8.16	13
FW3	51.71	61	23.47	201	77.28	112	12.57	20
FW4	53.47	87	24.22	639	101.31	187	22.24	27
FW5	51.65	61	23.95	182	106.61	142	19.83	25
IPC1	44.05	65	22.44	87	21.52	44	18.45	31
IPC2	32.34	33	17.14	41	8.36	9	6.08	8

5. Hash Table Reordering

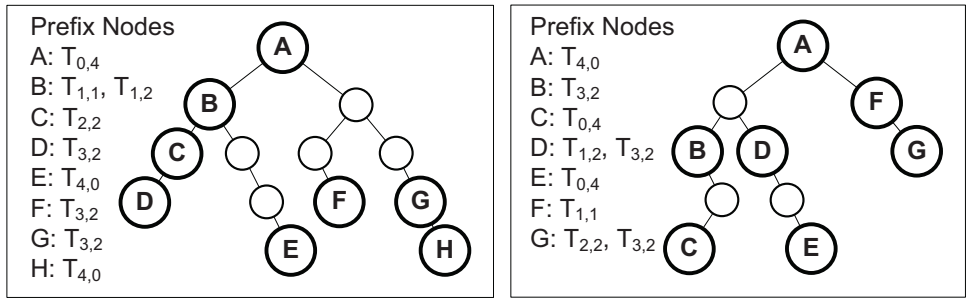
5.1. Algorithm

As mentioned above, *PTSS* is an advanced algorithm that avoids accessing tuples which do not match a query. However, the tuple lists do not delineate the precise relationship between filters and tuples. In the previous example, the search procedure for a packet header $(1010, 0010)$ would query $T_{2,3}$ since the best matching prefixes of both fields exist in $T_{3,2}$. Yet, there is no matching filters. The problem of *false match* would severely degrade the search performance of *PTSS*. In the worst case, no tuple is pruned.

By observing the search procedure of *PTSS*, we noticed that we can reduce the number of probed tuples by using the geometrical relations between two filters stored in tuple space. The relations of the two filters can be either non-overlapping, partially overlapping, or that one is a subset of the other. Two filters, F_a and F_b, are said to be non-overlapping if there is no packet matched to F_a and F_b simultaneously. In the case of partially overlapping filters,

Table 22. The tuples for the eleven filters in Table 1.

Tuple	Filter
$T_{0,4}$	F_1, F_2
$T_{1,1}$	F_3
$T_{1,2}$	F_4
$T_{2,2}$	F_5
$T_{3,2}$	F_6, F_7, F_8, F_9
$T_{4,0}$	F_{10}, F_{11}



(a) The prefix trie of f_1

(b) The prefix trie of f_2

Figure 5. Two prefix tries with the tuple list in the prefix nodes.

there exists at least one packet matched to F_a and F_b simultaneously while there also exists at least one other packet which only matches to F_a or F_b. In the last case, F_a is a subset of F_b if all packets matched to F_a also match to F_b. In this case, prefix f_i of F_b must be equal to f_i or a sub-prefix of f_i of F_a for all i, where $1 \leq i \leq d$ and f_i is the i_{th} prefix.

We use the third geometric relation that one filter is a subset of another filter to optimize the search procedure of *PTSS*. We start the search procedure from the hash table of the filters with longer prefixes. If there exists a matching filter, all the hash tables whose prefix lengths are shorter than or equal to those of the matching hash table can be pruned to reduce the number of hash tables to be accessed.

We illustrate our idea by using a set of 11 two-field filters in Table 1 as an example. We list *six* tuples in Table 22. Figure 5 shows the prefix tries of both fields, where each prefix node has a unique identifier corresponding to a tuple list. For an incoming packet with the header specification (0001,0110), the best matching tuple list of f_1 includes $T_{0,4}, T_{1,1}, T_{1,2}, T_{2,2}$ and $T_{3,2}$ and that of f_2 includes $T_{4,0}, T_{1,2}$ and $T_{3,2}$. By intersecting both tuple lists, *PTSS* probes $T_{1,2}$ and $T_{3,2}$ for possible matching filters. With our scheme, we can avoid probing $T_{1,2}$ since a matching filter F_6 is derived from $T_{3,2}$. Because F_6 is a subset of F_4, we store the action of F_4 in F_6 to ensure the accuracy of our scheme.

Next, we describe two construction procedures and the required data structure of our scheme. The first procedure sorts the tuples according to their prefix length combinations.

To make our scheme effective, we must start the search procedure of tuple accesses from the tuple with the longest prefix lengths. Therefore, we design a tuple-sort algorithm so that the tuple with longer prefixes can be accessed first. Initially, we create an empty tuple list to store the sorted tuples. Then, we calculate the value of $longer(T)$ for each tuple, where $longer(T)$ indicates the number of tuples whose corresponding prefixes are longer than or equal to those of T. For the example in Table 22, the value of $longer(T_{1,1})$ is *three* and those for $T_{0,4}, T_{3,2},$ and $T_{4,0}$ are *zero*. Next, we append the tuples whose values are *zero* to the sorted tuple list and remove these tuples from the unsorted tuple list. Then, we update the value of $longer(T)$ for the remaining tuples and repeat the above steps until all tuples are appended to the sorted tuple list. This procedure can guarantee that the tuples with longer prefixes usually have higher priority than those with shorter prefixes. This procedure requires at most $\|TS\|^3$ computation steps, where $\|TS\|$ denotes the size of tuple space (i.e., the number of tuples).

The second construction procedure calculates the actions that must be stored in each filter. This procedure starts from the first of the sorted tuples described above. For each tuple T, we use its filters to probe the tuples behind T. If there exists a matching filter in the probed tuple, we store the action of the matching filter in the processed filter. For single-match packet classification, we could store only the least cost action of the matching filters. This procedure requires $N\|TS\|$ computation steps, where N is the number of filters.

The data structure of our scheme includes one bitmap for each hash table. Each bitmap has $\|TS\|$ bits, and each bit corresponds to one tuple. For a tuple T, each bit of its tuple bitmap is set to *zero* if all the prefix lengths of the corresponding tuple are shorter than or equal to those of T. Otherwise, the bit is set to *one*. With the tuple bitmap, we can decide which tuples should be probed subsequently. To generate all tuple bitmaps, we need at most $\|TS\|^2$ comparisons.

The search procedure is basically identical to *PTSS* but with an extra comparison before accessing a tuple. The extra comparison determines whether the subsequently accessed tuple corresponds to a *zero* bit of the tuple bitmaps of the accessed tuples with a matching filter. If *yes*, the tuple access is avoided and the next tuple is considered.

5.2. Performance Evaluation

In this section, we use real and synthetic filter databases to investigate the performance of our scheme. The first part is a comparative evaluation by examining the performance of our scheme with the existing algorithms based on real filter databases. In the second part, we test the scalability of our scheme by using large synthetic databases.

The experimental results are listed in Table 23 and 24. As compared to *PTSS*, our scheme consumes slightly more memory space due to the extra tuple bitmaps. However, our scheme outperforms *PTSS* in terms of speed with the aid of tuple bitmaps and reordering. We believe that the extra storage is a reasonable tradeoff for speed improvement. For the other algorithms, our scheme outperforms *ABV* in most cases, but *HyperCuts* outperforms our scheme in most cases. *RFC* is the fastest algorithm, but it also consumes the most memory space.

We use 16K-filter synthetic filter databases for further evaluation. While our scheme is barely comparable to the existing algorithms for real databases, it is not the case for syn-

Table 23. Comparative Evaluation of Speed Performance Using Real Databases.

Databases	Filters	ABV	HyperCuts	PTSS	RFC	Our Scheme
ACL1	752	296.34	29.45	293.17	497.62	293.93
FW1	269	263.73	33.41	274.39	1,094.55	280.35
IPC1	1,550	331.46	142.18	299.15	8,984.18	306.42

Table 24. Comparative Evaluation of Speed Performance Using Real Databases.

	ABV		HyperCuts		PTSS		RFC		Our Scheme	
Databases	AMA	WMA	AMA	WMA	AMA	WMA	AMA	WMA	AMA	WMA
ACL1	35.13	44	14.79	22	20.18	32	11	11	17.52	25
FW1	20.49	30	21.37	46	85.25	124	11	11	19.73	47
IPC1	26.14	50	22.44	49	25.72	48	11	11	19.45	32

thetic databases. Table 25 shows that our scheme still consumes slightly more storage than *PTSS*, but it outperforms the other algorithms in most cases of storage performance. In addition, our scheme is the fastest among these algorithms, as shown in Table 26. Since the search performance of *ABV* is proportional to the number of filters and that of *HyperCuts* ties to the characteristics of filters, our scheme combines the storage efficiency of *PTSS* with efficient pre-computation to achieve a better balance between storage and speed performance.

Table 25. Comparative Evaluation of Storage Performance Using Synthetic Databases.

Databases	Filters	ABV	HyperCuts	PTSS	Our Scheme
ACL1	15,926	36,523.81	369.18	1,010.32	1,011.77
ACL2	15,447	55,141.81	1,359.68	2,055.93	2,062.33
ACL3	14,729	7,803.02	2,721.50	726.90	763.16
ACL4	15,405	13,306.42	1,985.26	793.31	838.88
ACL5	10,379	4,545.09	294.45	534.08	534.75
FW1	14,898	32,494.13	23,883.84	2,205.53	2,212.39
FW2	15,501	39,885.48	10,859.94	1,962.20	1,962.34
FW3	14,297	27,532.10	27,172.06	1,878.96	1,882.41
FW4	13,856	31,231.93	10,717.61	2,855.36	2,878.57
FW5	14,009	26,260.04	19,503.95	1,750.74	1,756.17
IPC1	14,954	16,116.21	3,557.90	716.50	732.50
IPC2	16,000	44,405.32	12,450.29	1,893.48	1,893.49

6. Conclusion

In this chapter, we present three algorithms for efficient packet classification based on hash tables. The first two algorithms combine hash-based algorithm with two complementary algorithms. These combinations are based on the observation that *Cross-producting* and *Bit Vectors* are complementary to hash-based algorithm in certain filter databases. This observation motivates us to develop two algorithms of filter categorization. For the combination with *Cross-producting*, the filters are either stored in a cross-product table or a tuple space.

Table 26. Comparative Evaluation of Speed Performance Using Synthetic Databases.

Databases	ABV		HyperCuts		PTSS		Our Scheme	
	AMA	WMA	AMA	WMA	AMA	WMA	AMA	WMA
ACL1	33.30	44	21.07	56	16.93	32	16.22	29
ACL2	36.55	50	21.72	132	28.49	57	13.78	34
ACL3	52.55	84	22.78	119	41.87	82	20.99	39
ACL4	50.00	84	21.78	92	39.03	77	21.32	38
ACL5	33.72	57	28.28	60	19.18	36	14.98	26
FW1	47.33	56	22.52	221	77.96	113	18.09	44
FW2	33.28	34	21.12	43	11.15	15	10.48	18
FW3	51.71	61	23.47	201	77.28	112	18.98	37
FW4	53.47	87	24.22	639	101.31	187	23.7	58
FW5	51.65	61	23.95	182	106.61	142	19.35	41
IPC1	44.05	65	22.44	87	21.52	44	19.47	36
IPC2	32.34	33	17.14	41	8.36	9	8.87	12

Our algorithm could avoid the overwhelming cross-product entries by limiting the size of cross-product table. For the combination with *Bit Vectors*, some filters are stored in the form of bit vectors to reduce the size of tuple space. As a result, our algorithms have much better feasibility than the existing algorithms. We also present the update procedures for both algorithms. The update procedures benefit from the compound data structures; thus, the cost of filter insertion and deletion can be minimized. In the third algorithm, we use the subset relations between two filters to reduce the number of probed tuples. Our algorithm starts by sorting the tuples based on their prefix length combinations, then we present the pre-computation for filters to ensure the correctness of packet classification. Our scheme uses the bitmap stored in each tuple to record the prefix-length orders among the other tuples. Our experimental results show that our algorithms could better balance speed and storage performance, especially for large synthetic filter databases. Therefore, the proposed scheme is suitable for network applications with numerous filters, such as firewalls and QoS routers.

References

[1] S Blake $et\ al$. An Architecture for Differentiated Services. *RFC*. 1998;(2475).

[2] Gupta P, McKeown N. Packet classification on multiple fields. In: *Proceedings of ACM SIGCOMM '99*; 1999. p. 147–160.

[3] Lianghong Xu YX Baohua Yang, Li J. *Packet Classification Algorithms: From Theory to Practice;* 2009. p. 648–656.

[4] Taylor DE. Survey and Taxonomy of Packet Classification Techniques. *ACM Computing Survey*. 2005;37(3):238–275.

[5] Kumar VP, Lakshman TV, Stiliadis D. Beyond best effort: router architectures for the differentiated services of tomorrow's Internet. *Communications Magazine, IEEE*. 1998 May;36(5):152–164.

[6] Srinivasan V, Varghese G, Suri S, Waldvogel M. Fast and scalable layer four switching. In: *Proceedings of ACM SIGCOMM '98;* 1998. p. 191–202.

[7] Baboescu F, Singh S, Varghese G. Packet classification for core routers: Is there an alternative to CAMs? In: *Proceedings of IEEE INFOCOM '03*; 2003. p. 53–63.

[8] Srinivasan V, Varghese G, Suri S. Packet classification using tuple space search. In: *Proceedings of ACM SIGCOMM '99;* 1999. p. 135–146.

[9] Gupta P, McKeown N. Packet Classification using Hierarchical Intelligent Cuttings. In: *Proceedings of Hot Interconnects VII;* 1999. .

[10] Sumeet Singh GV Florin Baboescu, Wang J. Packet Classification Using Multidimensional Cutting. In: *Proceedings of ACM SIGCOMM '03;* 2003. p. 213–224.

[11] Gupta P, McKeown N. Algorithms for packet classification. *IEEE Network Magazine.* 2001;15(2):24–32.

[12] Baboescu F, Varghese G. Scalable Packet Classification. In: *Proceedings of ACM SIGCOMM '01*; 2001. p. 199–210.

[13] van Lunteren J, Engbersen T. Fast and scalable packet classification. *IEEE Journal on Selected Areas in Communications.* 2003 May;21(4):560–571.

[14] Lakshminarayanan K, Rangarajan A, Venkatachary S. Algorithms for advanced packet classification with ternary CAMs. In: *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications;* 2005. p. 193–204.

[15] Liu H. Efficient Mapping of Range Classifier into Ternary-CAM. In: *Hot Interconnects X*; 2002. p. 95–100.

[16] Woo T. A modular approach to packet classification: algorithms and results. In: *Proceedings of IEEE INFOCOM '00*; 2000. p. 1213–1222.

[17] Lu W, Sahni S. Packet Classification Using Space-Efficient Pipelined Multibit Tries. Computers, *IEEE Transactions on.* 2008;57(5):591–605.

[18] Pao D, Liu C. Parallel tree search: An algorithmic approach for multi-field packet classification. *Computer Communications.* 2007;30(2):302–314.

[19] Srinivasan V. A packet classification and filter management system. In: *Proceedings of IEEE INFOCOM '01*; 2001. p. 1464–1473.

[20] Lakshman TV, Stidialis D. High-speed policy-based packet forwarding using efficient multi-dimensional range matching. In: *Proceedings of ACM SIGCOMM '98*; 1998. p. 203–214.

[21] Sun X, Sahni SK, Zhao YQ. Packet classification consuming small amount of memory. *IEEE/ACM Transactions on Networking.* 2005;13(5):1135–1145.

[22] Wang PC. Scalable packet classification with controlled cross-producting. *Computer Networks.* 2009;53(6):821–834.

[23] Taylor DE, Turner JS. Scalable Packet Classification using Distributed Crossproducting of Field Labels. In: *Proceedings of IEEE INFOCOM '05;* 2005. p. 269–280.

[24] Zheng K, Liang Z, Ge Y. Parallel packet classification via policy table pre-partitioning. In: *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE.* vol. 1; 2005. p. 73–78.

[25] Eben-Chaime M, Mehrez A, Markovich G. Capacitated location-allocation problems on a line. *Computers and Operations Research.* 2002;29(5):459–470.

[26] Wang PC, Lee CL, Chan CT, Chang HY. Performance improvement of two-dimensional packet classification by filter rephrasing. *IEEE/ACM Transactions on Networking.* 2007;15(4):906–917.

[27] Kounavis ME, Kumar A, Vin H, Yavatkar R, Campbell AT. Directions in packet classification for network processors. In: *Proceedings of Second Workshop on Network Processors (NP2);* 2003. .

[28] Miguel A Ruiz-Sanchez EWB, Dabbous W. Survey and Taxonomy of IP Address Lookup Algorithms. *IEEE Network Magazine.* 2001;15(2):8–23.

[29] Sahasranaman V, Buddhikot MM. *Comparative evaluation of software implementations of layer-4 packet classification schemes;* 2001. p. 220–228.

[30] Song H. *Evaluation of Packet Classification Algorithms, http://www.arl.wustl.edu/ hs1/PClassEval.html* ;. .

[31] Taylor DE, Turner JS. Classbench: a packet classification benchmark. In: *Proceedings of IEEE INFOCOM '05;* 2005. p. 2068–2079.

[32] Lampson B, Srinivasan V, Varghese G. IP lookups using multiway and multicolumn search. *IEEE/ACM Transactions On Networking.* 1999 June;7(4):323–334.

[33] Wang PC, Chan CT, Hu SC, Lee CL, Tseng WC. High-speed packet classification for differentiated services in ngns. *IEEE Transactions on Multimedia.* 2004;6(6):925–935.

[34] Wang PC. Scalable Packet Classification with Hash Tables. *IEICE Transactions on Communications*. 2010;E93-B(5):1155–1158.

[35] Wang PC. Scalable Packet Classification Using a Compound Algorithm. *International Journal of Communication Systems.* 2010;23(6-7):841–860.

[36] Wang PC. Efficient Packet Classification with a Hybrid Algorithm. *IEICE Transactions on Information and Systems.* 2009;E92-D(10):1915–1922.

INDEX

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y