Optical Flow Switching with Time Deadlines for High-Performance Applications

by

Anurupa Ganguly

B.S., MIT (2008)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2009

Author .....
Department of Electrical Engineering and Computer Science
May 7, 2009

Certified by .....
Vincent W.S. Chan
Joan and Irwin Jacobs Professor of Electrical Engineering & Computer Science
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Optical Flow Switching with Time Deadlines for High-Performance Applications

by

Anurupa Ganguly

Submitted to the Department of Electrical Engineering and Computer Science
on May 7, 2009, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis focuses on the design and analysis of scheduling approaches for Optical Flow Switching (OFS) serving high performance applications with very stringent time deadline constraints. In particular, we attempt to meet setup times only slightly longer than one roundtrip time with networks at moderate to high loading. In this work, we propose three possible scheduling mechanisms for OFS connection setup in a WDM network: (i) a simple algorithm, which awards pre-emptive priority to applications requiring time deadline performance; (ii) a multi-path probing mechanism using only coarse average loading information (i.e., no detailed network state information) but without pre-emption; and (iii) a multi-path probing mechanism using periodically updated network state information and without pre-emption. The updating scheme calls for a slow control plane, which refreshes and broadcast network states only periodically on the order of seconds or longer. Our results show that for a low blocking probability, the update interval must be a fraction of the mean service time of transactions. We conclude that this algorithm, a combination of both slow centralized and fast distributed processes, delivers an efficient and scalable control design for a high-speed transport network of the future.

Thesis Supervisor: Vincent W.S. Chan
Title: Joan and Irwin Jacobs Professor of Electrical Engineering & Computer Science

Acknowledgements

I would like to thank my advisor Professor Vincent Chan for his continuous guidance and support over the last two years. It has truly been an honor to have had the opportunity to work with such a gifted researcher. I am indebted to him for all his efforts in not only making me stronger technically, but also for making me a better communicator and writer.

I would also like to thank Guy Weichenberg for the countless hours he spent with me in defining and working through my research problem.

Last, but not least I would like to thank my parents, whose love and encouragement has been the foundation of my education at MIT.

The research in presented in this thesis has been supported by:

- Defense Advanced Research Projects Agency, "Future Optical Network Architectures"
- National Science Foundation, "Future Internet Design"
- CISCO

Contents

List of Figures

Chapter 1

Introduction

[1] With the development of Wavelength Division Multiplexing (WDM) technology, today, an optical fiber carries up to hundreds of wavelength channels, each supporting data rates up to 40Gb/s. One of the most attractive features of optical networks is this abundance of bandwidth in optical fiber. However, appropriate design of the network architecture for optical networks has not developed nearly as fast as advancements in physical layer technology. Unfortunately until recently, architectures geared for electronic communications were still used for optical communications, which has severe drawbacks. This is due to the following noteworthy differences between electronics and optics [12]: (a) a cost-effective and feasible optical hardware components such as Random Access Memory (RAM) have not yet been developed, (b) optical technologies are still too expensive given the fact that the current cost of an optical logic gate is at least 10 orders of magnitude higher than that of an electronic logical gate, and (c) the fundamental limit of minimum switching energy of an optical logic gate is much higher than that of electronics; therefore more power is dissipated for the same number of logical operations. We anticipate future increases in data volume per transaction (two to three orders of magnitude) due to growing demands for high-bandwidth applications. For larger transaction sizes, scalability

[1]The following sections through section 1.3 has been taken from [30]

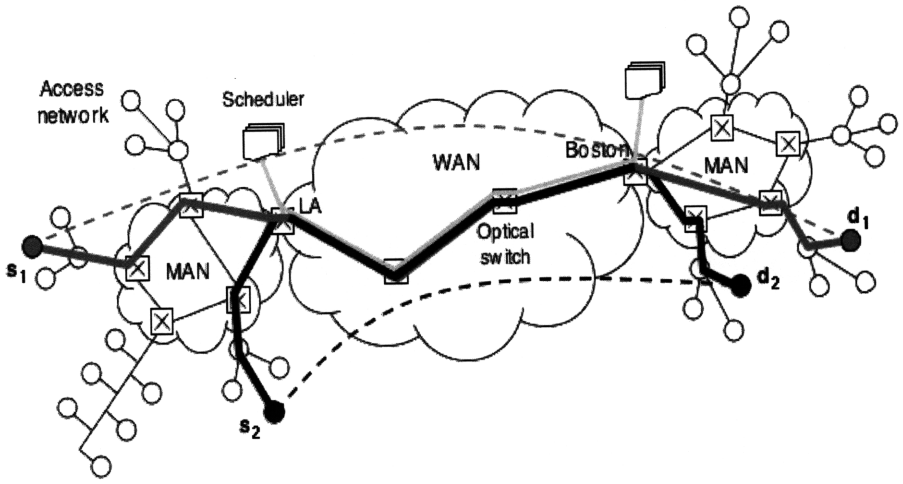

Figure 1-1: OFS with transparent, end-to-end data flow between users. [10]

of electronic network architectures becomes a constraint. This can be overcome with optics due to the large capacity of fiber and WDM technology as stated above. However, it also calls for a shift in architectural designs for high-speed optical networks.

In recent works [6,10-19, 21-23,25-29], these authors explored the use of an optical network transport mechanism, Optical Flow Switching (OFS), in the quest for lowering network cost by the same order of magnitude as anticipated increase in transaction lengths. Although the concept of OFS was introduced long ago in [11], due to the lack of demand for such high-bandwidth communications, the performance, implementation, region of operation, and cost of OFS were not studied in-depth until recently. OFS is a scheduled, all-optical, end-to-end service in which connections are established in response to flow-based requests by client-layer IP network schedulers (e.g., routers) for direct access by individual users. To use network resources efficiently, service holding times of wavelength channels are required to be on the order of hundreds of milliseconds or longer. Scheduling of flows with a time horizon of several transactions also help to achieve high network utilization albeit with some queueing delay at the

entrance of the network usually in the form of holding the user from transmission until the network is ready for the transaction, [29]. However, there are specialized applications that require stringent time deadlines and would not like to wait in a queue but would be willing to pay more to use the network as soon as possible. In this work we explore a fast OFS implementation for stringent time deadline services. This fast flow switching architecture is highly efficient and economical for on-demand high data rate transfers, distributed sensor data ingestion, as well as distributed computing and processing with short time deadlines and bursty, high-volume data transactions. A good example application of OFS is when the optical network is used as a service-bus in a Service Oriented Architecture (SOA) context that has four types of users: (1) data gatherers, (2) distributed storage devices, (3) computing and processing servers, and (4) unscheduled users who need access to data and processed information. Other examples are where optical networks are used as the transport for grid computing and cloud computing. While in previous works [25-27] we focused on network physical architecture and cost of providing the OFS transport mechanism, this work addresses a specialized class of OFS service that provides very fast setup times ($\sim$ one roundtrip delay) with no queueing. In contrast to optical burst switching, there is no collision due to contention, and thus this fast service does not need back-offs and retransmissions that results in long tails in the delay distribution, unless the network is very over provisioned and way under-utilized so no collisions ever occur. We will describe an online algorithm and its optimization to make this fast OFS service viable. The following are the main architectural ideas addressed in this thesis:

1. A baseline centralized but slow control plane for efficient network utilization operating in a time scale of seconds and longer
2. A fast wavelength service on a virtual overlay network for bursty applications with time deadlines that uses a hybrid combination of centralized (slow processes) control and distributed (very fast) control that can set up

sessions faster than 100 ms (one roundtrip time plus hardware switching time of a few ms).

3. Mixing different classes of service for efficient network operations: fast service will coexist with longer duration, book-ahead, and best-effort services.

OFS is an all-optical networking approach with a distinct feature that sets it apart from architectures such as optical packet switching (OPS) and optical burst switching (OBS): it enables a connection-oriented network providing all-optical connections between users with two-way end-to-end reservations. In OFS, users employ an off-band signaling protocol to request lightpaths for their transactions, and the network schedules a dedicated, end-to-end lightpath for the duration ($>$ 100ms transaction times) of the transfer thereby preventing collisions due to contention. When the transaction finishes, the network resources are relinquished to other users. Each transaction is scheduled based on a time delay requirement over a finite time horizon. One of the merits of OFS is that it allows large transactions to bypass intermediate electronic routers in the backbone. This eliminates intermediate router processing ((including access and metro-router processing), a key idea behind a traditional IP routing paradigm. Efficient, dynamically assigned multi-access broadcast groups can be arranged for multiple transaction durations using a particular node architecture [27].

There will be a tradeoff among three observable network performance parameters: delay, blocking probability, and wavelength utilization. This tradeoff allows the multiple levels of service quality to co-exist in the same network. The key to high utilization of backbone wavelength channels a precious network resource owing to the necessity of optical amplifiers and dispersion management is statistical multiplexing of large flows from many users in a scheduled fashion. Thus, high network utilization can be achieved if the users are willing to wait for service according to a schedule (incurring delay) or accept high blocking probability upon request for service, [29]. Figure 1-2 shows the increase in uti-

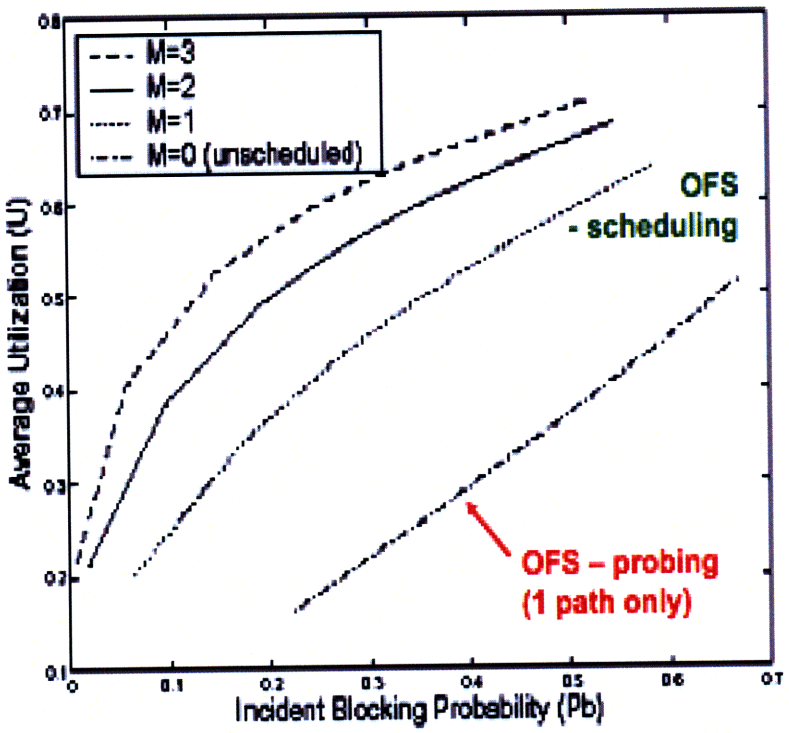

Figure 1-2: OFS with and without scheduling horizons, [29], showing increased utilization with increasing scheduling horizon (M transactions) for the same blocking probability performance. [30]

lization for the same blocking performance as the scheduling horizon increases. However, there are critical services that may desire the economy of OFS but not willing to accept the waiting time of a scheduled service. Thus, in this work we show how to provide fast OFS setup with little more than one round-trip time from user to user and still uses a network with high loading. [30]

1.1 OFS Control Architecture

Scheduling and connection setup for OFS is a complex problem with many different implementation schemes. Connection setup for flows may be handled using both distributed or centralized scheduling. An inherent tradeoff exists between centralized and distributed scheduling schemes. Centralized schemes, by virtue of having complete state information and global coordination, always provide solutions that are no worse than those of distributed schemes. On the other hand, distributed schemes are scalable for large networks and more resilient than centralized schemes. In a typical network setting, there exists widely varying QoS requirements for data: some sessions require setup times of no longer than 100 ms, while other sessions can tolerate setup times on the order of several seconds. We will consider a hybrid scheduling framework comprising both centralized and distributed components, whose complementary advantages allow us to gracefully support these heterogeneous traffic requirements with scalability.

For sessions requiring sub-second setup times, routing and wavelength assignment (RWA) must be completed and the session initiated immediately. Though centralized approaches yield network configurations at least as good as those of distributed schemes, the propagation and computation times involved, in view of the stringent session setup requirements, will be barely feasible, if at all. Thus, global network coordination on fast sub-second time scales (< 1 s) is nei-

ther scalable nor easy to implement. Moreover, we anticipate that some future applications can benefit from even faster setups (~5-10 ms). We therefore propose a hybrid centralized (slow processes) /distributed (fast processes) scheme relying on up-to-date local and slightly stale global information to setup these sessions for a class of special users over a virtual overlay subnet of the optical network.

1.2 Background

As broached in the introduction, this shift back towards traditional circuit-switched networks employed in OFS leads us to investigate scheduling schemes that integrate both centralized and distributed processes. Connection setup in the WAN has been studied previously in numerous literatures. [1] presents a next generation optical network architecture termed Generalized Multiprotocol Label Switching (GMPLS). Although GMPLS requires all-optical connection setup in the WAN as OFS, it differs in the LAN and MAN design, making it a router-to-router reconfiguration approach. In [1], the authors conclude network reconfigurations on the order of minutes in order to maintain feasible control packet overhead.

[2] and [4] describe a scheduling algorithm in which the user reserves a lightpath based on network information that is periodically updated. Both papers report that these implementations reduce blocking probability for small update intervals, however, may result in stale network information for reasonable control traffic and delay. [3] studies a scheme with updates that are based on triggered mechanisms rather than periodic ones. Here, nodes from time to time, as a result of a large amount of change in link status as observed by these nodes, update the network state. Due to the nature of these updates, [3] showed that this approach also results in stale network information as well as incurs very

high delays of updates.

1.3 Candidate Algorithms

We will present and analyze three different setup algorithms for OFS. To simplify the traffic model in the following three algorithms, we divide all traffic in the network into two classes: through traffic and cross traffic. Through traffic is initiated by the originating node of a source-destination node pair. Cross traffic is traffic in any of the internal links of the network of all other flows not originated from the source node (i.e. excluding the first link out of the source node). In a general setting, end users would have buffering capability for large transactions, however maintaining certain delay constraints. In this work, because we must deliver negligible setup delays, buffering capability is not considered in the analysis.

1. **Algorithm 1**: This algorithm gives through traffic pre-emptive priority over cross traffic. From the perspective of through traffic users, cross traffic is not present in their designated paths. Therefore, knowledge of network state at an originating node of through traffic is accurate at all times since the only relevant state information is the occupied links that have been previously assigned by the same source node. In contrast to the following two approaches, this algorithm does not require probing but is clearly very disruptive to cross traffic and thus has dubious utility. It is included for comparison purposes and as a upper bound of the efficiency of this fast OFS service.

2. **Algorithm 2**: Algorithm 2 allows through traffic and cross traffic to occupy links without any priority as in Algorithm 1. Upon a through traffic arrival at a source node, the user/source-node probes all paths between the source and destination to choose an available path. Probing all paths

yields the best performance for such probing schemes but presents a large burden for the network management system and is often wasteful. It is included here as a bench mark.

3. **Algorithm 3**: In this algorithm through traffic has no pre-emptive priority. Network state (link-states) information is periodically broadcasted (a slow process) to all nodes by a central network manager. The algorithm also uses a distributed approach in which the source node sends out multiple pre-computed (slow and centralized) path requests (lightpath probes: a fast localized process) to the nodes residing on these multiple paths. These path requests may contain preferences based upon path lengths and slightly stale global network information. If the network states are updated and broadcast on time scales less than the shortest session durations, the state information is likely to be fairly accurate. The algorithm selects the smallest number of paths and the best (based on predetermined preferences) set of paths to probe to satisfy the target blocking probability.

For either Algorithm 2 or 3, the algorithm dictates the control network data rate and time elapsed between network state broadcasts . Furthermore, a request for service along a path may or may not explicitly include a particular subset of wavelength channels on which to possibly transmit. Nodes on a path may suggest any of the open wavelength available along the path. If the resources requested at a node are (not) available, then the node forwards an ACK (nothing) to the next node along the path en route to the destination. In the event that two or more simultaneous requests for the same resources arrive at a node, the node adjudicates the dedication of the resources according to the priorities of the requests, how many nodes each lightpath request has successfully passed through up to that point (longest survivor may have priority everything else being equal), as well as its own version of slightly stale global network state information. Immediately after forwarding an ACK, a path node temporarily reserves the relevant resources in case the source/destination nodes choose to

use these resources for the session . In the event that multiple ACKs have arrived at the destination node, each corresponding to different lightpaths, the destination node decides which path to use. Upon making this decision, the destination node notifies the source node of the chosen path, and the source node begins data transmission immediately thereafter. Simultaneously, the destination node sends release messages along all lightpaths that have ACKed but will not be used for data transmission, to release these network resources. Also, the centralized management system is notified so that it will refresh its lists of available resources in the next update. In this fashion, lightpaths can be set up in as little time as one roundtrip plus processing and hardware-reset times ($\sim$5 ms).

For Algorithm 3, it is important to discuss the role and limitations of the control plane, which manages and sets up end-end connections in the WAN. In this hybrid-scheduling scheme, the network manager in the control plane is responsible for broadcasting network state updates to all users at a constant interval. For a dynamic OFS network, the network state changes very rapidly for moderate traffic due to different classes of users. Intuitively, smaller update intervals would decrease blocking probability. However, If the update interval needed to preserve high-performance is less than the sum of the roundtrip propagation delay, processing delay of updates at the end user, and computation time of state updates, then the network state information used to probe is inaccurate with high probability. In this case, even very small update intervals may lead to incorrect network information. The control plane is handled electronically; therefore updating user nodes too frequently also has the possible drawback of excess overhead of control packets. Given data rate limitations of electronics, such a control mechanism may not be scalable for highly frequent updates. There is also the obvious question of feasibility of network configurations at this rapid rate, which again is subject to the data rate restrictions and processing power of the control plane.

The contribution of this work is two-fold: First, the design and analysis of an optimal probing algorithm that seeks to minimize the effects of inaccurate information so as to reduce blocking and setup failure. Secondly, we will present an analysis of the tradeoff between the interval at which the network state is updated and viability of such a scheduling algorithm in comparison to the other two approaches. Network operators would want to maximize utilization of network resources by all users to minimize the overall cost, while ensuring ultra-high performance for the end user. Hence, the question of resource-sharing and pre-emption of certain traffic types becomes critical in our study - specifically determining whether priority should be given to applications requiring such high-reliability service, as opposed to adding scheduling complexity of maintaining a centralized control manager. Each subsequent section will illustrate the performance of the three scheduling algorithms. In the last chapter we offer a comparison of all three to assess which is most suitable for high-performance OFS applications with minimum setup delay.

1.4 Applications

The main applications for such high-reliability OFS implementations would be defense networks whose frontier is in network centric military operations. The US Department of Defense (DoD) initiated the Global Information Grid (GIG), which was designed primarily to support very high traffic loads [6]. For example, video communications of battlefields around the world, on-demand to all operational sites, is one such military application that requires a robust network architecture with very high performance and bandwidth requirements, with negligible setup times. This same concept of video-on-demand also has various commercial applications such as downloading high-definition videos on the Internet - which could take hours given our current IP standards. OFS also has applications in other forms of Internet traffic. [7] and [8] have shown

that Internet traffic has a heavy-tailed distribution. In other words, much of the traffic volume of Internet traffic comes from large transactions. If OFS supports these transactions, then processing delays at intermediate IP routers can be alleviated to a great extent. Lastly, there are tremendous applications in the biomedical field. These include integrated views of MRIs and other scans between physicians in different locations and on demand delivery of these scans for more effective and rapid patient-care. Another important application is video-surgeries, which could come into the forefront in the near future for routine procedures.

Chapter 2

Scheduling Approaches without Centralized Control

2.1 Algorithm 1

The first design we will consider is modeled in Figure 2-1, where source s and destination d are connected by K_s total paths each with h links. In the general case, the number of links per path is characterized by $H = \{h_1, h_2, ...h_{K_s}\}$, where h_i is the hop-size of path i. To simplify our analysis, Figure 2-1 depicts the case where $h_i = h_j \ \forall i, j$. In this algorithm, the only traffic is through traffic between the source-destination pair. The traffic at node s is modeled as a single Poisson process of rate λ_t and exponentially distributed transaction times with service rate μ. Because of the priority structure in this algorithm, from the through traffic user's perspective, paths are never occupied by cross traffic. Therefore, there is no need to probe channels before transmitting and the source is always aware of which paths are being used. Upon a flow request at s, one path is chosen at random from those available. All links along that path are reserved immediately for the duration of the flow. Therefore, there is only blocking in this case when all K_s paths are servicing through arrivals. We examine the blocking probability as a function of the number of total paths

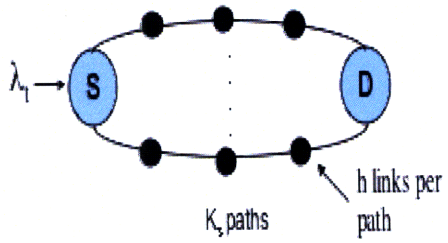

Figure 2-1: Algorithm 1: Through traffic pre-empts all other traffic internal to the paths except for those originating from the same source node. Traffic arrives as a Poisson arrival process at source node s destined for node d with K_s total paths, each with h links, available.

K_s between source-destination, and the offered load λ_t. Using the Erlang B formula, the blocking probability for a $M/M/k/k$ queue ($k = K_s$) is

$$P_B = \frac{\rho_t^{K_s}/n!}{\sum_{n=0}^{K_s} \rho_t^n/n!}$$

where $\rho_t = \lambda_t/\mu$ is the utilization constant. Figure 2-2 shows the number of paths required given a desired blocking probability of 10^{-2}, for various network loads ρ_t. Note the load on the horizontal axis is normalized to that of a single wavelength.

Since through traffic in this model can pre-empt cross traffic, the performance is as though there is no cross traffic present in the network. This sets a lower bound on the minimum number of paths needed between any source-destination pairs for no pre-emption and cross-traffic being allowed at each path. The obvious observation is that for low blocking probability, the paths must be fairly lightly loaded or loaded mostly with pre-emptable traffic. This is a reality for this type of fast flow switching with hard time deadlines. In other words, if network economy is an issue, the network must also serve pre-emptable scheduled flows as

a second class service. By the average connectivity of a US backbone network, the mean node degree is around 2.6. This limits the number of independent paths that may be probed per source-destination pair. In addition, probing requires reservation of candidate paths for the roundtrip propagation time plus the transaction duration. For this reason path utilization also becomes a factor, as excess paths should not be tied up for a single flow request. Therefore, for manageability and also to facilitate easy network topology design, the number of paths needed should be less than approximately 5. Therefore, the total offered loading $\lambda_t t$ in Figure 2-2 should be upper-bounded by ~ 1.4 (normalized load per path ~ 0.3).

Since there are K_s paths, the loading per path is $\frac{\rho_t}{K_s}$. Therefore, the utilization of each path is given by $\frac{\rho_t}{k} - [P_B * \frac{\rho_t}{K_s}]$. Path utilization as a function of offered load is shown in Figure 2-3 for $P_B = 10^{-2}$. The paths are clearly underutilized as, even if we double the offered load, the path utilization hardly goes up by 10 percent. Although this model serves as a best-case scenario from the performance perspective of through arrivals, it does not utilize channel resources very efficiently. When the network is heavily loaded with through traffic, cross traffic will be often pre-empted.

2.2 Algorithm 2

2.2.1 Algorithm 2a: without correlation amongst links

In this approach (Figure 2-4), through traffic from s to d as well as cross traffic at the links of each path are both present. This implementation differs from the previous one in that through traffic has equal priority as cross traffic. To gain some analytical insights, we have simplified the statistical model to assume the traffic arriving at each link is independent of one another with $p_{i,j}$ being the probability that link j of path i is serving a transaction. Figure 2-4 depicts the special case when $p_{i,j} = p \quad \forall i, j$. An arrival from s to d will be blocked if

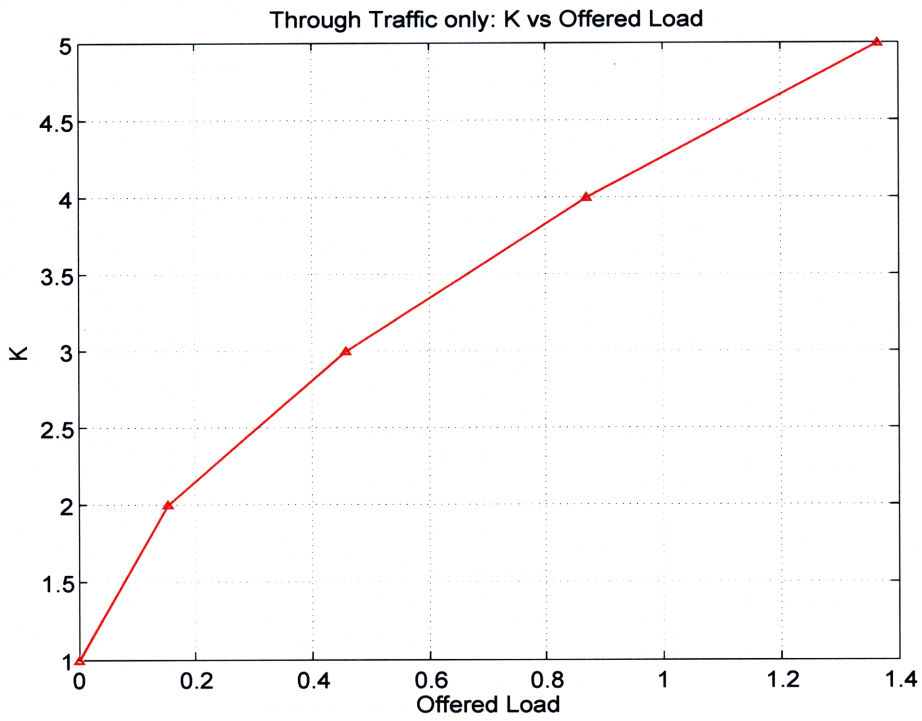

Figure 2-2: Number of paths K_s needed to maintain $P_B = 10^{-2}$, as a function of offered loading ρ_t, for Algorithm 1. An offered load of 1 corresponds to one wavelength of data.

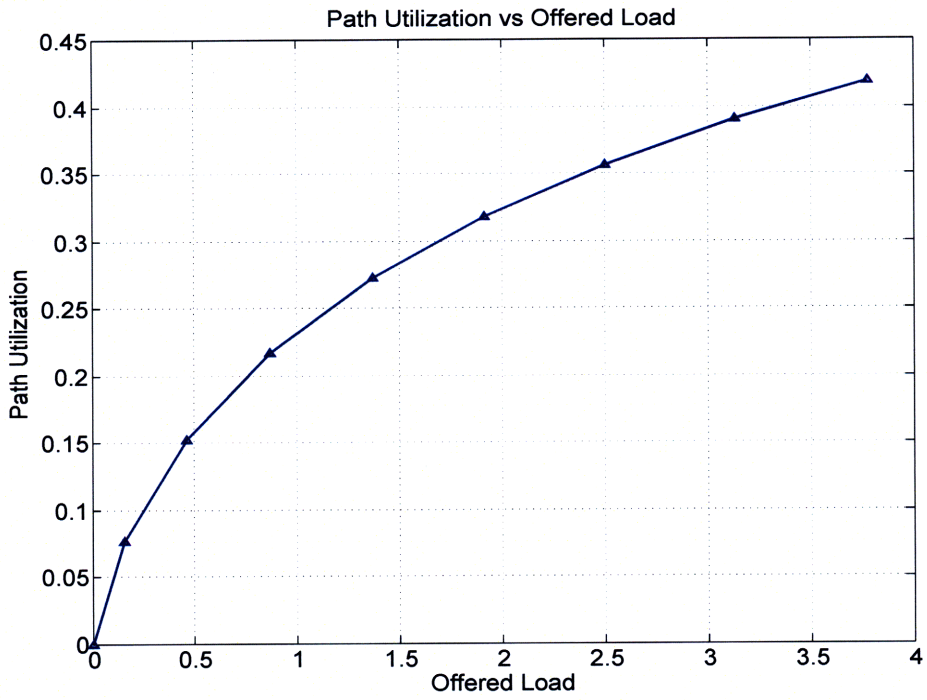

Figure 2-3: Path utilization ($\frac{\rho_t}{k} - [P_B * \frac{\rho_t}{K_s}]$) as a function of offered load for $P_B = 10^{-2}$, for Algorithm 1. An offered load of 1 corresponds to one wavelength of data.

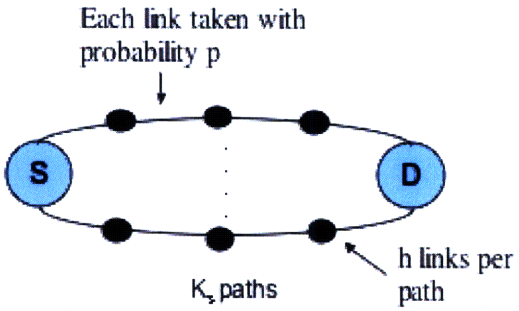

Figure 2-4: Algorithm 2: Through traffic does not pre-empt cross traffic. Probability a link is occupied is denoted by p. K_s total paths, each with h links.

at least one link of each path is being used on all K_s paths. Because through traffic no longer pre-empts cross traffic, the source node no longer has complete knowledge about the availability of paths, as they may be occupied by cross traffic. Here we must note that although each path has h_i links respectively, the source has full knowledge of the status of the first link. Therefore, we assume that each path in Figure 2-4 in actuality has $h+1$ links; however the first link does not factor into our computation since its state is known to the source. To enhance the performance of through arrivals under high network loading conditions, the end user probes all or a substantial fraction of all possible paths to find an available path before transmitting. The expression for the blocking probability of the general case with different hop-sizes per path is:

$$P_B = \prod_{i=1}^{K_s} \left\{ 1 - \prod_{j=1}^{h_i} (1 - p_{i,j}) \right\} \tag{2.1}$$

Assuming that all paths have equal hop-size ($h_i = h_j \ \forall i, j$), equation 2.1 may be simplified to:

$$P_B = (1 - (1-p)^h)^{K_s} \Rightarrow K_s = \frac{\log(P_B)}{\log(1 - (1-p)^h)} \tag{2.2}$$

Using the approximation that $\log(1+\epsilon) \approx \epsilon$ for small ϵ, the number of paths as a function of p, P_B, and h is:

$$K_s = \frac{-\log(P_B)}{(1-p)^h} \tag{2.3}$$

By equation 2.2, Figures 2-5, 2-6, and 2-7 show K_s as a function of p for the blocking probabilities $10^{-1}, 10^{-2}$, and 10^{-4}. Each subplot shows the values of K_s for hop sizes 1-10. The required number of paths quickly increases beyond feasible limits for high loading and hop-size. To best utilize network resources, we do not ideally want to probe more than five paths. The tradeoff between values of K_s, p and h is given in Figure 2-8. It offers an upper bound on p per link for $P_B = 10^{-2}$ such that K_s does not exceed 10 for both the actual relationship (2.2), and the first approximation (2.3). For algorithm 2, we observe that p must be very low to achieve a desirable blocking probability of 10^{-2} for multi-hop paths.

If we further approximate $(1-p)^h \approx (1-hp)$, then (2.3) becomes:

$$K_s = \frac{-\log(P_B)}{(1-hp)} \tag{2.4}$$

Furthermore, with the previous approximation from our expression in (2.2), we get:

$$K_s = \frac{-\log(P_B)}{\log(hp)} = -log_{hp}P_B \tag{2.5}$$

Figure 2-9 shows these three approximations of K_s in comparison to the exact expression for various values of p. As expected, for large p (greater than 0.5), the first approximation is a good approximation. When the network is very lightly loaded, ($p < 0.15$), only the third approximation estimates the value of K_s accurately.

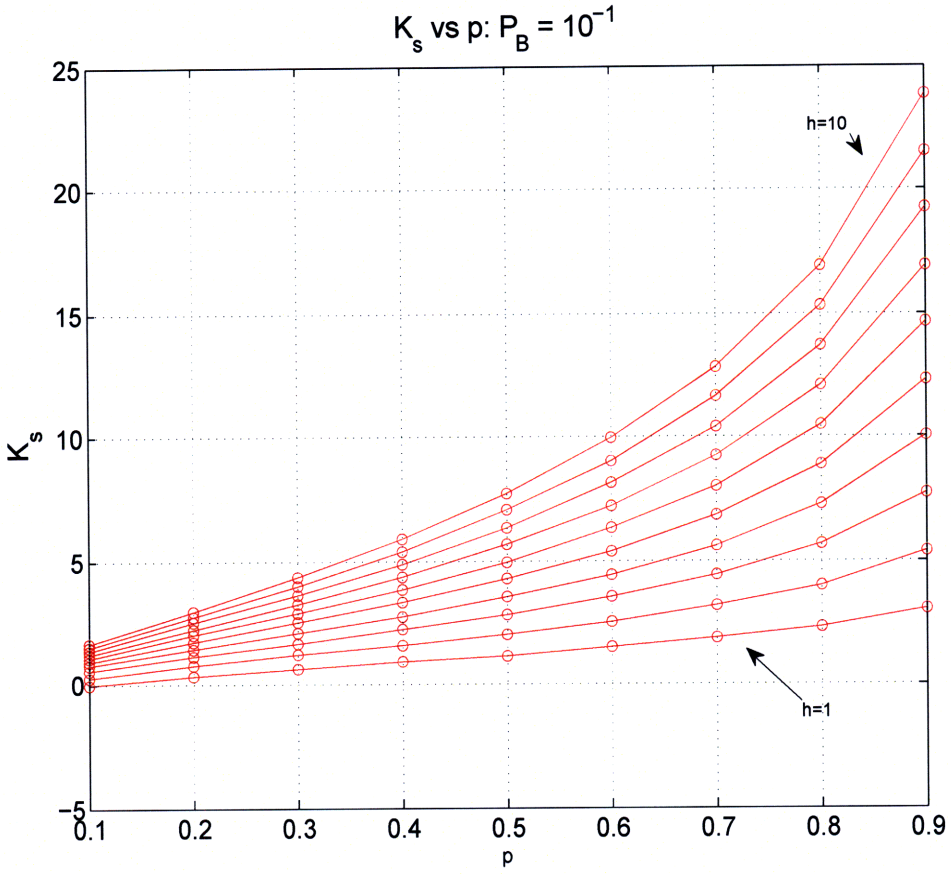

Figure 2-5: Semi-log plot of K_s as a function of p for $P_B = 10^{-1}$ and hop sizes (h) 1-10 for Algorithm 2a.

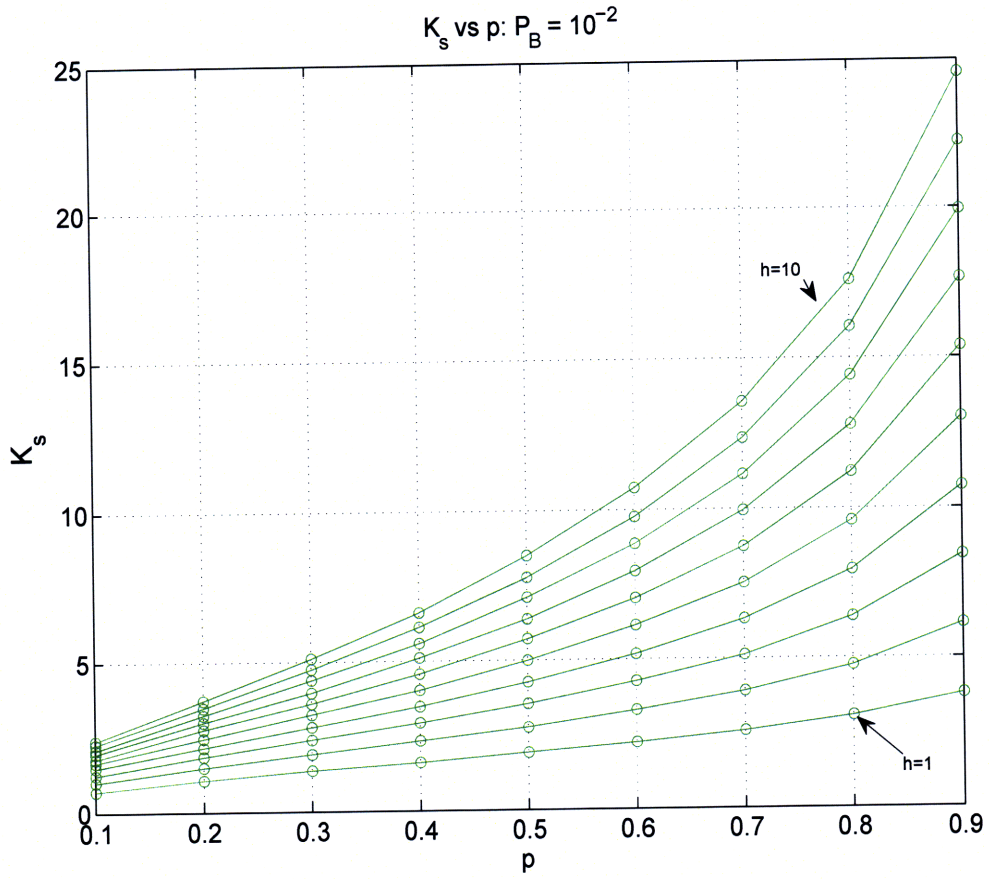

Figure 2-6: Semi-log plot of K_s as a function of p for $P_B = 10^{-2}$ and hop sizes (h) 1-10 for Algorithm 2a.

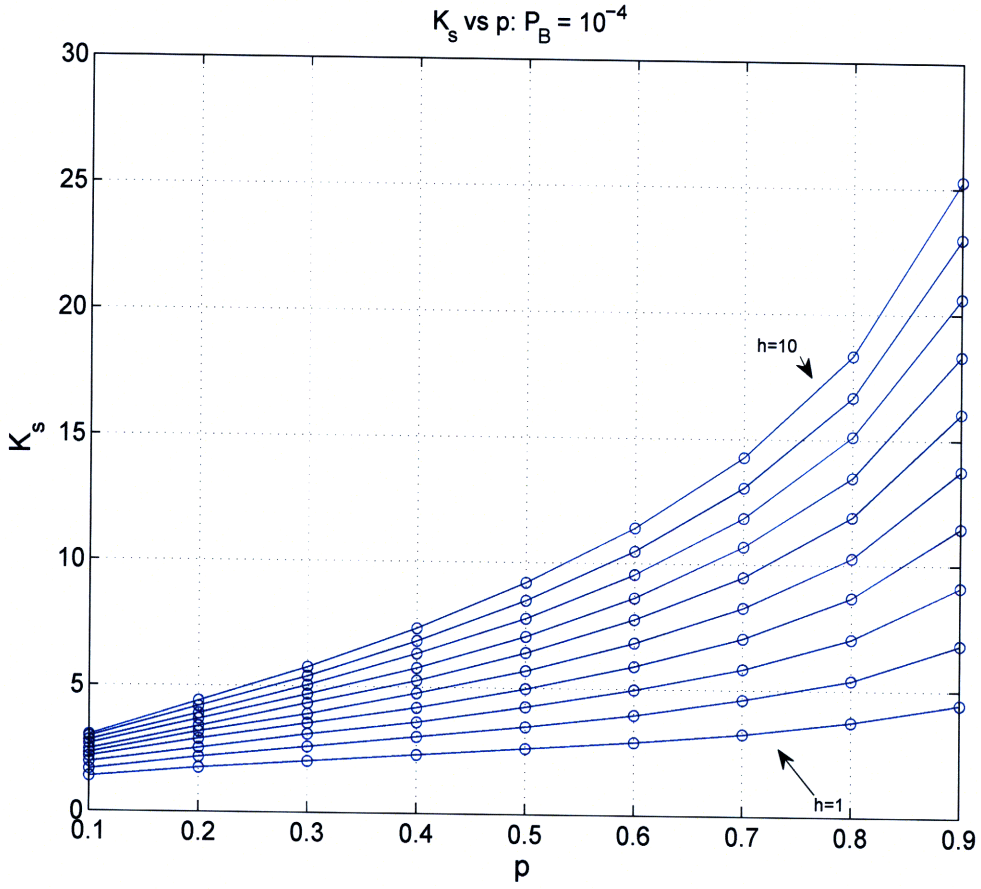

Figure 2-7: Semi-log plot of K_s as a function of p for $P_B = 10^{-4}$ and hop sizes (h) 1-10 for Algorithm 2a.

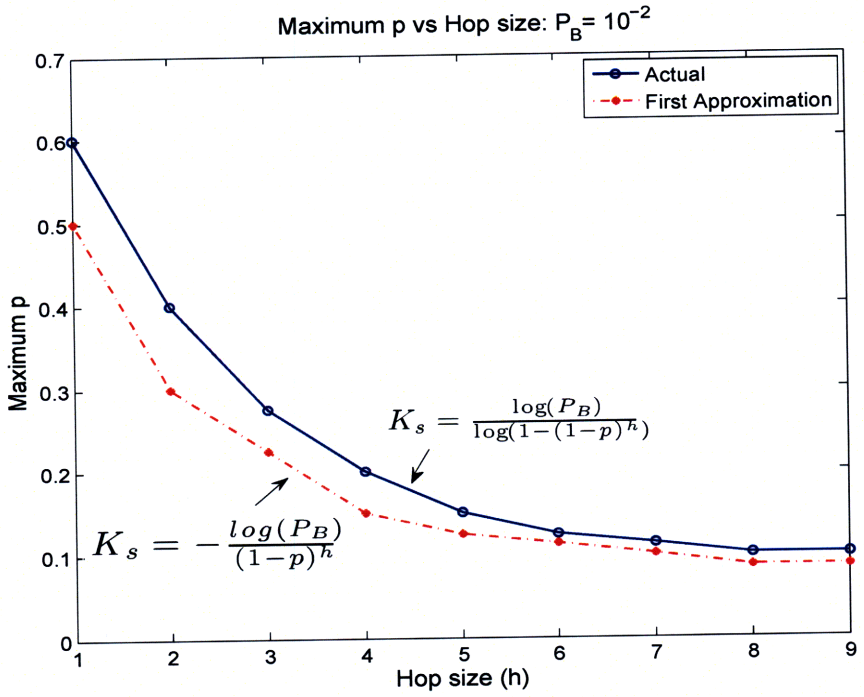

Figure 2-8: Upper bound of p versus hop size (h) for Algorithm 2a. It includes both the actual expression and the first approximation. The maximum value of p was found such that $K_s = 10$. $P_B = 10^{-2}$

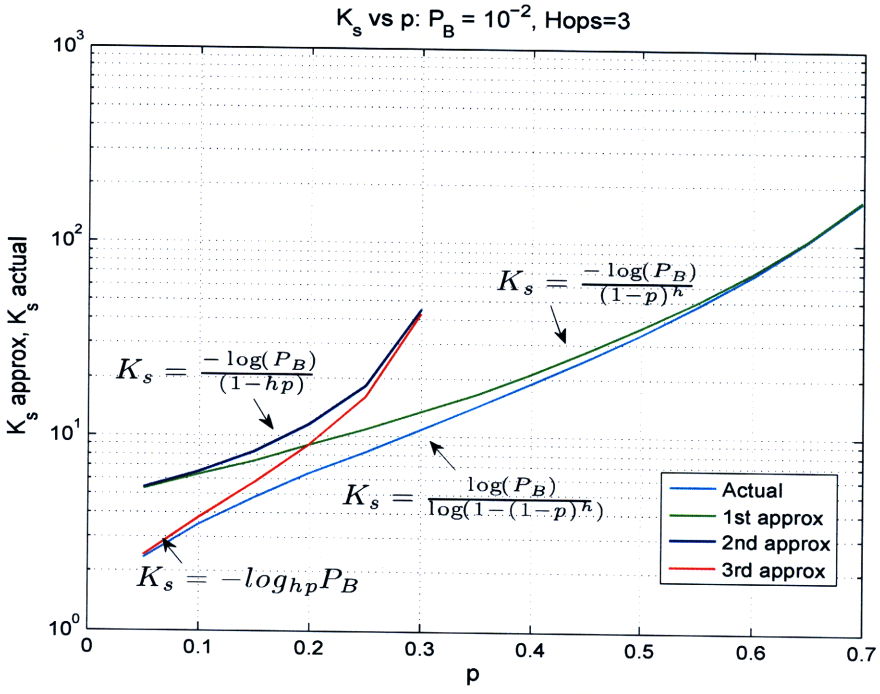

Figure 2-9: Three approximations and actual value of K_s as a function of offered load for Algorithm 2a. Hop Size = 3, $P_B = 10^{-2}$

One of our objectives is to analyze how the presence of cross traffic affects the performance of through traffic. This can be done through a comparison of the respective K_s values required by Algorithm 1 and Algorithm 2 under the same required blocking probability. In order to study the difference in K_s for the two cases, the offered load per link must be the same. For Algorithm 1, we assumed that the probability a link is occupied is p whereas in Algorithm 1, the offered load per link was $\frac{\rho_t}{K_s}$.

Let us also define the packet arrival process into each link for Algorithm 2 as Poisson process with rate λ_c and exponential service rate μ. The offered load into each link must be $\rho_c = \frac{\rho_t}{K_s}$ to match the loading per link of Algorithm 1. Any link will be unavailable if there is already one arrival being served. In this way, each link may be modeled as a $M/M/1/1$ queue. Thus, the probability that the link is unavailable is $\frac{\rho_c}{\rho_c+1}$. By relating $p = \frac{\rho_c}{\rho_c+1}$, we get a relationship between K_s and the offered load per link λ_c.

Figure 2-10 shows K_s values for Algorithm 2 (denoted as K_{CT}) as a function of K_s values for Algorithm 1 (denoted as K_{TT}). We see that the values of K_s for Algorithm 2 rapidly increase as hop-size increases. It is apparent that allowing cross traffic into the network for paths that has more than a couple of hops has a very detrimental effect of needing much more paths between node pairs and coupled with a corresponding hit on lowering allowable loading of the network. The main reason for this poor performance is the wavelength continuity constraint, which requires the same wavelength for all links along the path. The more links there are in the path, the more likely the wavelength is blocked for the same average loading. It is justifiable to argue that for moderate to high loading, allowing cross traffic without priority for through traffic is not a feasible design. The following section conducts the same investigation of

required K_s values, but this time with correlation between connected links.

2.2.2 Algorithm 2b: with correlation amongst links

The previous approach assumes independence of traffic between subsequent links, but let us now assume that there is correlation amongst links along a path. In other words, the event that a link j is occupied is dependent on the state of link $j-1$. We denote l_j as the event where link j is free, ρ as the correlation coefficient of the states of two connected links, and p as the probability that a single link is occupied as before. In [9], the author shows that the conditional probability a link is unoccupied given the previous link is unoccupied, is the following:

$$P(l_j|l_{j-1}) = \rho(p) + (1-p) \tag{2.6}$$

As shown in the previous section, an arrival at s will be blocked if any of the links along a path are taken. Therefore, the blocking probability is:

$$P_B = (1 - [(1-p)(\rho * p + (1-p))^{h-1}])^{K_s}$$

$$K_s = \frac{\log(P_B)}{\log((1 - [(1-p)(\rho * p + (1-p))^{h-1}]))} \tag{2.7}$$

Figures 2-11 to 2-16 show the relationship between K_s values for this modified algorithm, denoted as K_{CT}, versus K_s values for Algorithm 1, denoted as K_{TT}. The values of ρ are 0.2, 0.5 and 0.9, respectively and the offered load is again equalized for the two models ($\rho_c = \rho_t/K_s$). As the correlation coefficient approaches 1, this translates to the fact that cross traffic transactions span a larger number of links. In effect, the traffic resembles that of Algorithm 1, which span all h links. Therefore as the correlation increases, we see a dramatic decrease in the value of K_{CT}. For $\rho = 0.9$, the values of K_{CT} actually fall below the values

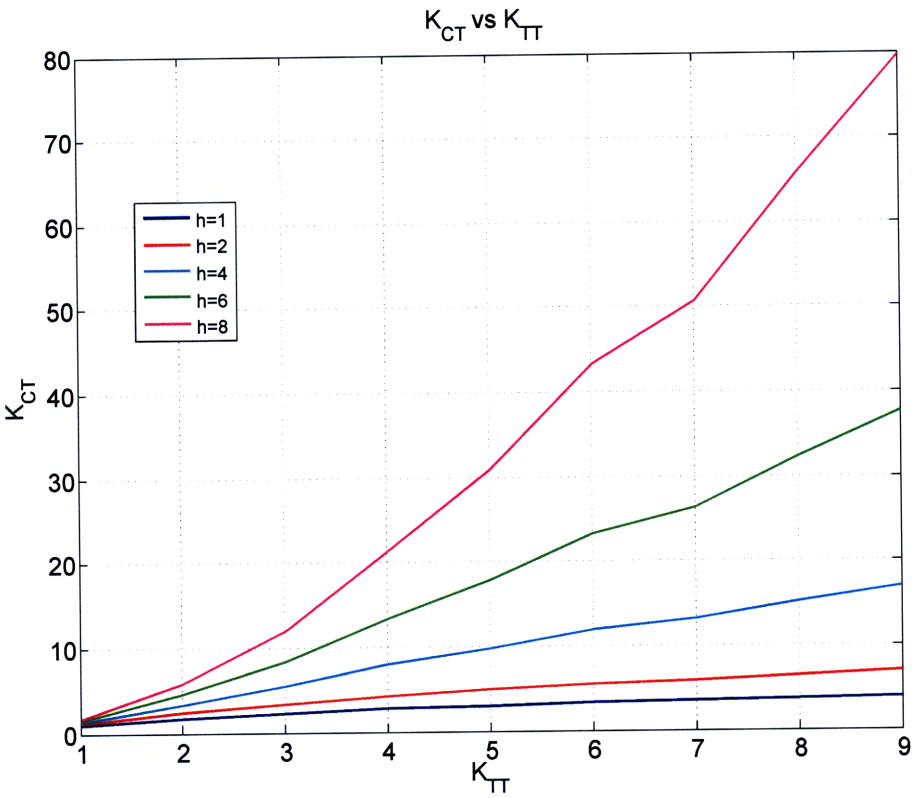

Figure 2-10: Required number of paths for Algorithm 2a, K_{CT} as a function of required number of paths K_{TT} for Algorithm 1 for the same offered load and $P_B = 10^{-2}$.

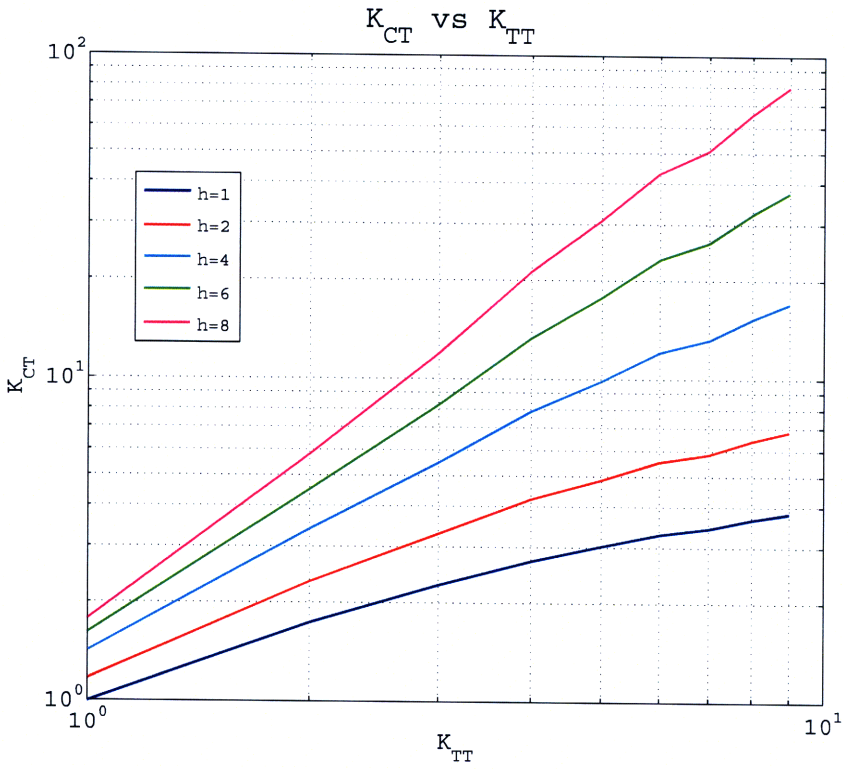

Figure 2-11: Required number of paths for Algorithm 2a, K_{CT} as a function of required number of paths K_{TT} for Algorithm 1 for the same offered load and $P_B = 10^{-2}$ in log-scale.

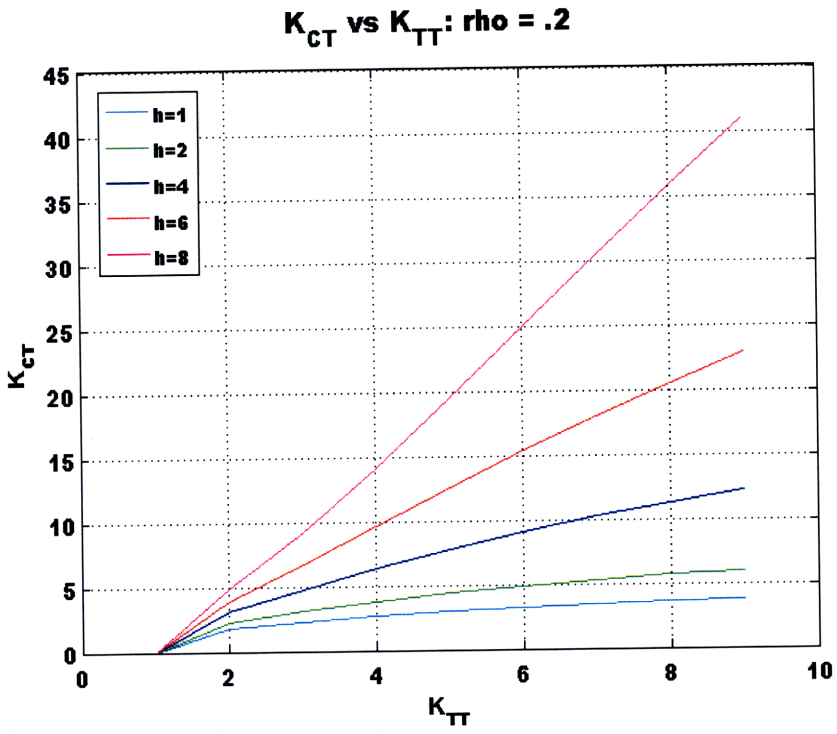

Figure 2-12: Required number of paths for Algorithm 2b, K_{CT} as a function of required number of paths K_{TT} for Algorithm 1 for the same offered load, $P_B = 10^{-2}$ and $\rho = 0.2$.

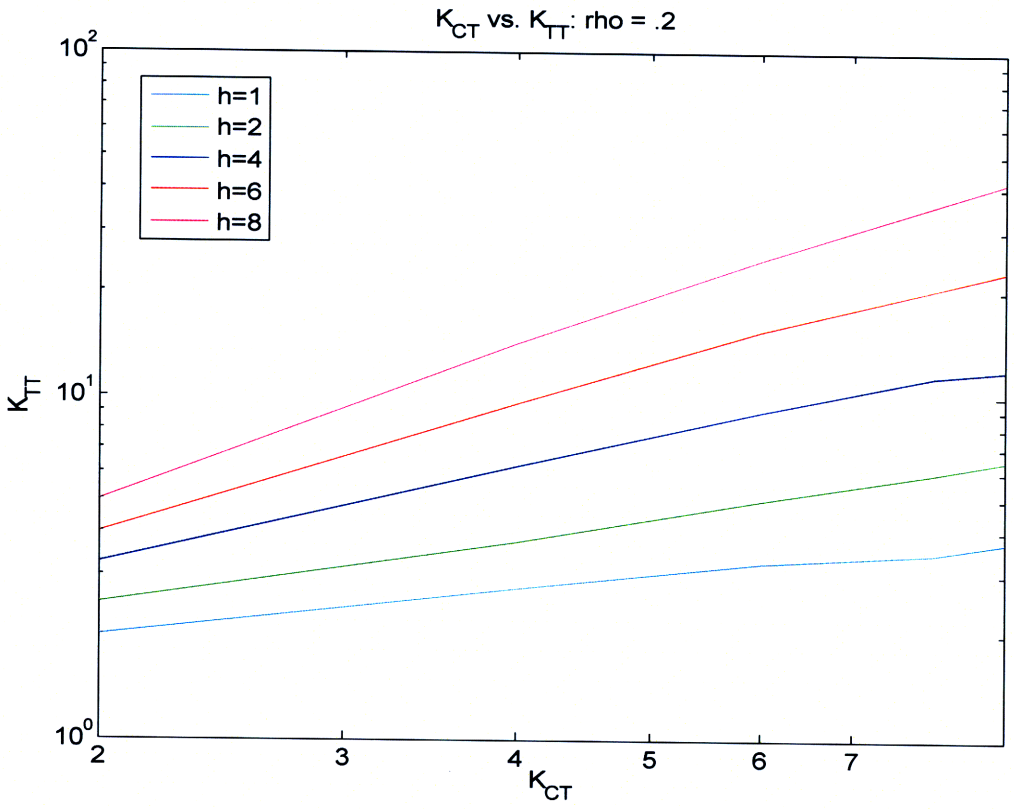

Figure 2-13: Required number of paths for Algorithm 2b, K_{CT} as a function of required number of paths K_{TT} for Algorithm 1 for the same offered load, $P_B = 10^{-2}$ and $\rho = 0.2$ (log-scale).

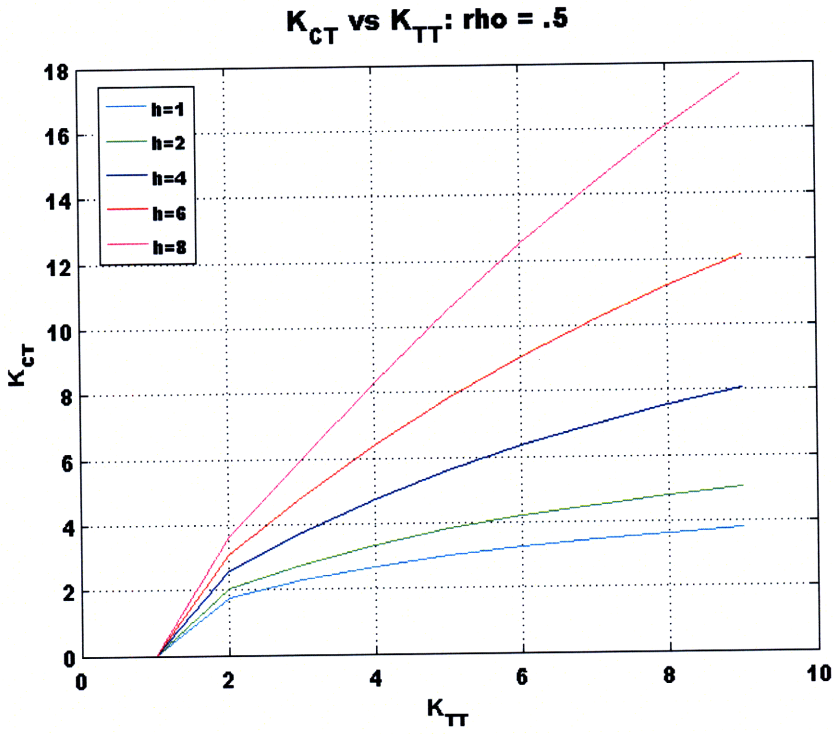

Figure 2-14: Required number of paths for Algorithm 2b, K_{CT} as a function of required number of paths K_{TT} for Algorithm 1 for the same offered load, $P_B = 10^{-2}$ and $\rho = 0.5$.

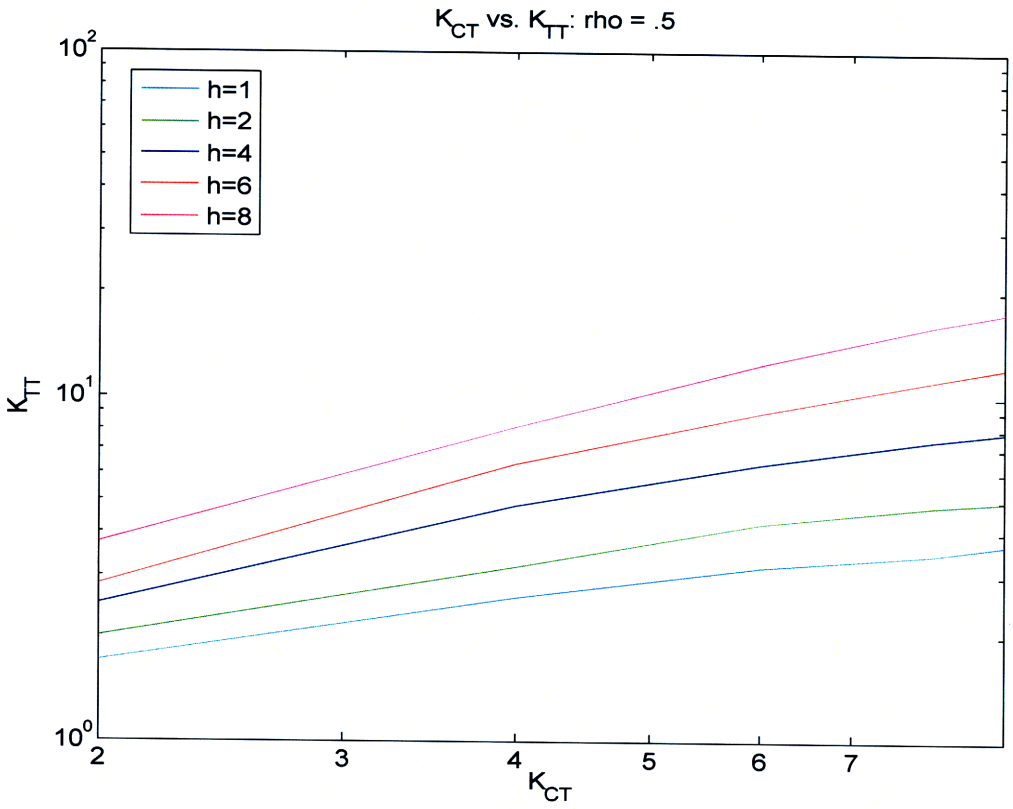

Figure 2-15: Required number of paths for Algorithm 2b, K_{CT} as a function of required number of paths K_{TT} for Algorithm 1 for the same offered load, $P_B = 10^{-2}$ and $\rho = 0.5$ (log-scale).

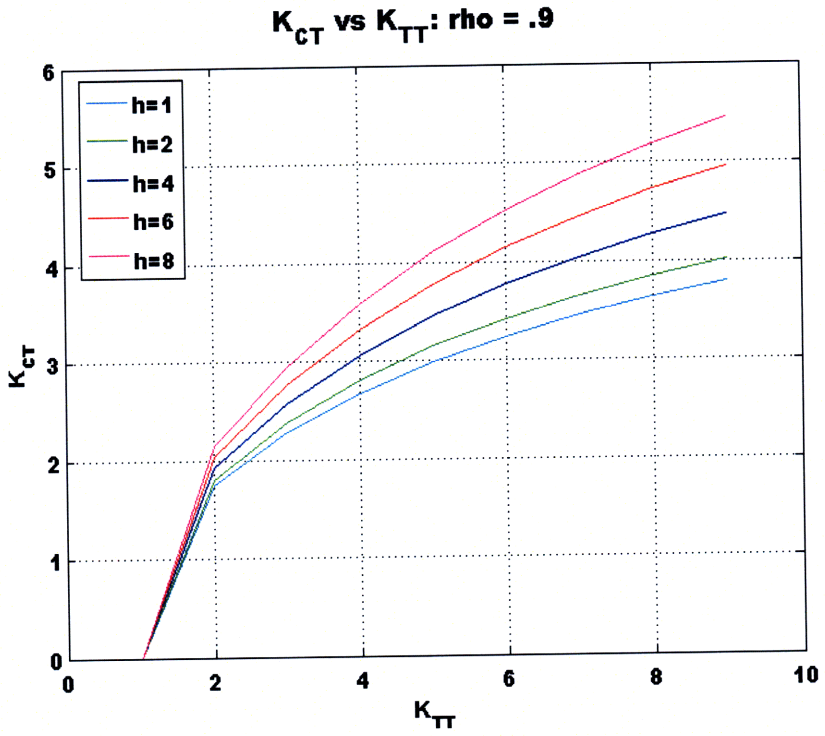

Figure 2-16: Required number of paths for Algorithm 2b, K_{CT} as a function of required number of paths K_{TT} for Algorithm 1 for the same offered load, $P_B = 10^{-2}$ and $\rho = 0.9$.

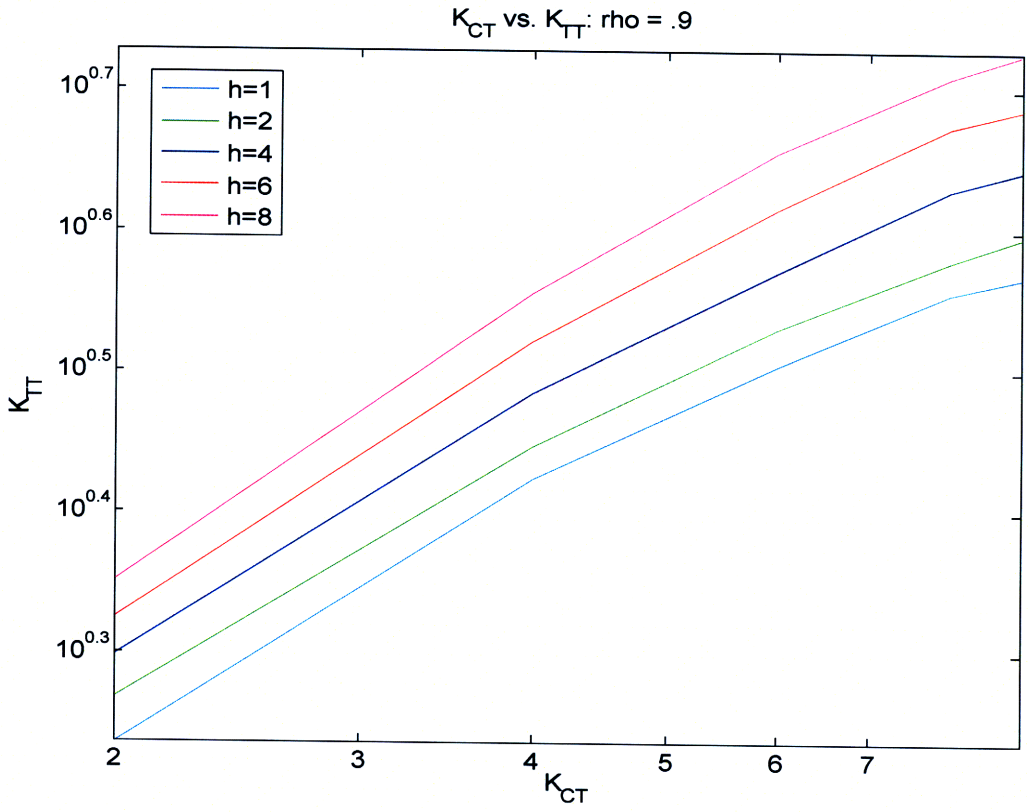

Figure 2-17: Required number of paths for Algorithm 2b, K_{CT} as a function of required number of paths K_{TT} for Algorithm 1 for the same offered load, $P_B = 10^{-2}$ and $\rho = 0.9$ (log-scale).

of K_{TT}, however this does not translate to better performance for Algorithm 2. In order to better understand this trend, let us consider the extreme case where $\rho = 1$. When $\rho = 1$, the traffic per path resembles through traffic since all traffic originating at the source node is carried on all h links. Let us for example think about an arrival into link 3 of path 1. If path 1 is blocked, even if another path is free to carry the transaction, the arrival will be blocked. For the same intensity of through and cross traffic, a higher proportion of cross transactions are blocked in the presence of open paths since they are constrained to only one path. Since the occupancy of cross traffic is lower for the same rate, blocking probability decreases for through arrivals. As a result, the number of required paths also drops off. Algorithm 3, discussed next, is a hybrid system with periodic network state updates to significantly reduce probing efforts. [30]

Chapter 3

Scheduling Approach with Centralized Update Mechanism

3.1 Algorithm 3

As in Algorithm 2, through traffic and cross traffic in this system design share paths without any priority assigned. This algorithm uses a hybrid scheduling approach that employs a centralized scheduler to periodically update all node pairs on the currently occupied paths. Unless the network state has just been updated and the exact state of each path is known, to serve new arrivals, the source node still needs to probe multiple paths to achieve a low blocking probability for a heavily loaded network. However, with the additional information of the updated network state, the number of paths to be probed will be smaller than that required for Algorithm 2.

There are two types of traffic at each link as shown in Figure 3-2. The first is through traffic, an uninterrupted flow, between s and d. Second, cross traffic, at each intermediate link of a path between s and d, that has originated from source nodes of other source-destination pairs. Both through and cross traffic in this paper are characterized by independent Poisson processes with arrival rates λ_t and λ_c, respectively, and expected service rate μ. For simplicity of

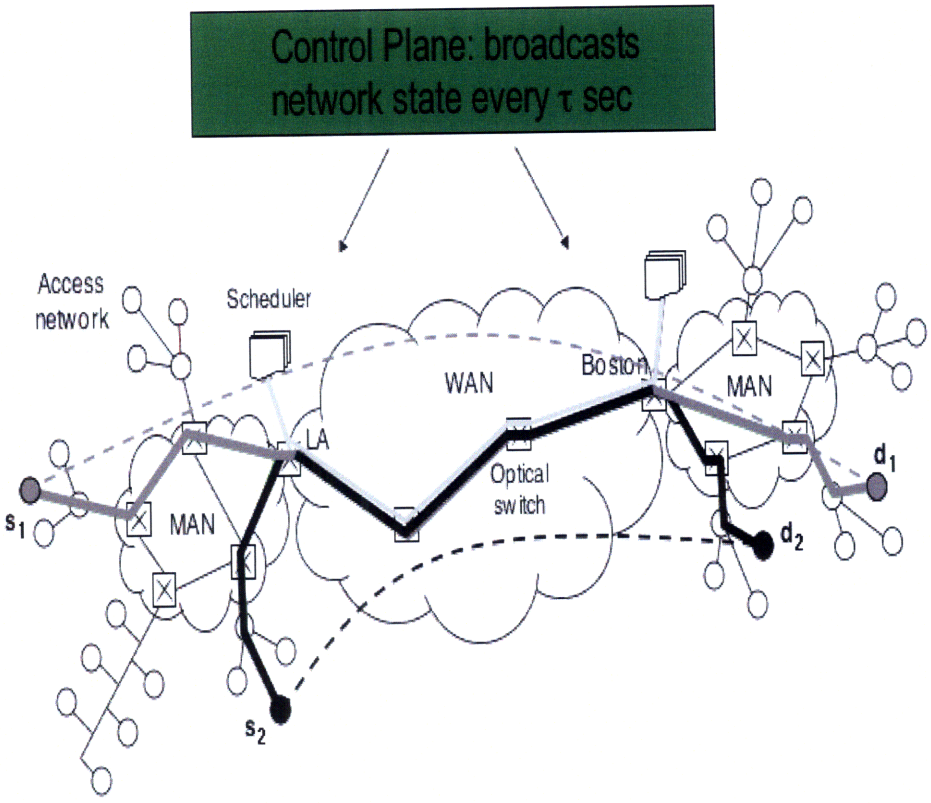

Figure 3-1: Model 3: Centralized OFS architecture with update mechanism [10]

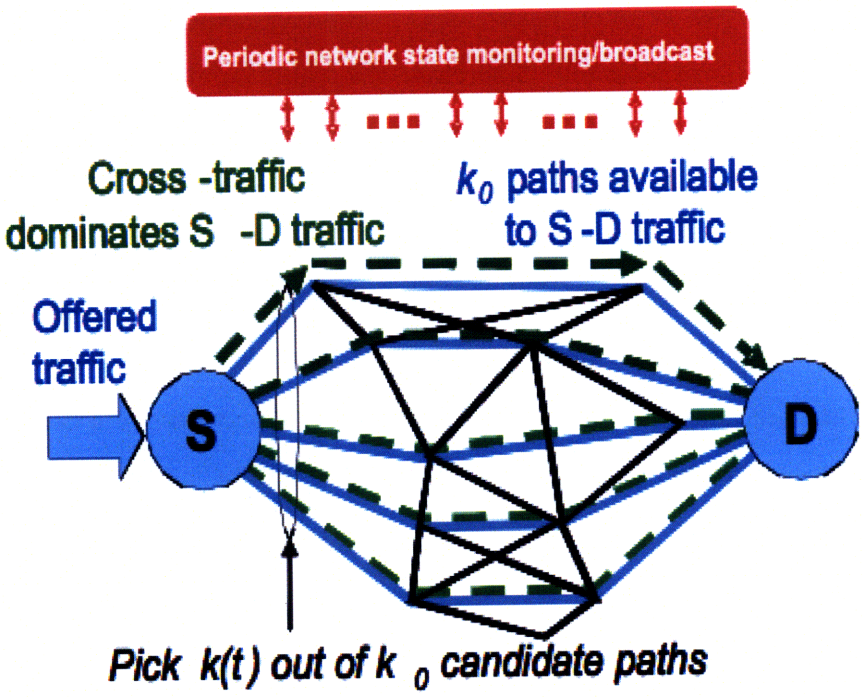

Figure 3-2: Algorithm 3: The (slow) centralized network management system periodically broadcasts the path states at regular intervals. The user/scheduler probes $K_p(t) = min\{K_k, K_o(t)\}$ paths to achieve the desired blocking probability. [30]

analysis, these traffic types are assumed to be statistically independent though there are actually subtle dependencies. The scheduling manager broadcasts the updated network state periodically at interval τ to all user nodes. We would like to determine the longest possible update interval that achieves acceptable throughput-blocking performance.

At each network update, the scheduling manager identifies the set of paths that are busy, {set of busy paths: $\mathcal{K}_A$}, therefore also the set of paths that are available, {set of open paths: $\mathcal{K}_B$}. The cardinalities of sets $\mathcal{K}_A$ and $\mathcal{K}_B$ are K_a and K_b respectively. Therefore, the total number of paths between s and d, $K_s = K_a + K_b$. Upon a flow request from s to d at time t, s must use the network state information from the most recent broadcast to choose the optimum set of paths to probe, $\mathcal{K}_P(t)$. One can easily see that at the update, $\mathcal{K}_P(0) = \mathcal{K}_B$. The number of paths in the set $\mathcal{K}_P(t)$ is $K_p(t)$. If all $K_p(t)$ paths are unavailable upon probing, the flow is blocked. For any $t > min\{1/\mu, 1/\lambda_c\}$ (time greater than the minimum of the average service time of a flow, and average inter-arrival time) the network information will become stale since without a broadcast update the source node no longer has accurate information of the traffic at the internal links of the previously announced open paths. Thus we predict τ must be $< min\{1/\mu, 1/\lambda_c\}$.

3.1.1 Analysis of Blocking Probability

Case 1: $\tau \ll 1/\mu$

We begin by defining the parameters referred to in this section:

1. K_s: the total number of paths between source-destination pair
2. K_b: the number of paths available at the last update
3. K_a: the number of path occupied at the last update

4. $K_T(t)$: the subset of K_b paths taken by through traffic at time t
5. $K_o(t)$: $K_o(t) = K_b - K_T(t)$ is the remaining subset of K_b that may be open or occupied by cross traffic at time t
6. $K_p(t)$: The number of paths that are probed as a function of time elapsed since the most recent update
7. K_k: Fixed probing threshold - If $K_o(t) < K_k$, then $K_p = K_o$. Otherwise, $K_p = K_k$.
8. $P_{C|k}(t)$: Given $P(K_o = k)$, this is the probability that a path in the subset k has been taken by a cross arrival since the last update
9. τ: the update interval
10. $\beta(t)$: the blocking probability of a through arrival
11. ρ_t: the offered load due to through traffic. $\rho_t = \lambda_t/\mu$
12. ρ_c: the offered load due to cross traffic. $\rho_c = \lambda_c/\mu$
13. μ: the service rate of transactions

Specifically, Algorithm 3 operates according to the following procedure. The value of K_k is pre-computed to achieve a given desired blocking probability. At time t, if $K_o(t) < K_k$, then all $K_o(t)$ paths are probed ($K_p(t) = K_o(t)$). If $K_o(t) > K_k$, only K_k paths (selected at random) are probed ($K_p(t) = K_k(t)$). Blocking of an arrival occurs when all $K_p(t)$ paths probed are occupied. The approximations used in the computation of $\beta(t)$ are summarized below:

Assumptions

(1) Network state information is broadcasted to all user nodes with interval τ. This implies that when a user requests a flow setup, at most τ time units have passed since the last update. Therefore it is likely that all paths in $\mathcal{K}_A$ are still occupied by the assumption that τ is a lot smaller than $1/\mu$. Thus, while

choosing $\mathcal{K}_P$, we will only consider $\mathcal{K}_B$, the paths that were free at the most recent update.

(2) Since we would like to design an algorithm with a small blocking probability even for a heavily loaded network, we make the additional approximation that all through arrivals will be carried, which is optimistic but not excessively so since we are interested in blocking probabilities of the order of 10^{-2} or lower. As a result, the distribution of the number of paths taken by through traffic, $K_T(t)$ may be approximated by a Poisson random variable:

$$P_{K_T}(k_T) = \frac{(\lambda_t t)^{k_T}(e^{-\lambda_t t})}{k_T!}$$

(3) From our assumption that update intervals are fractions of the expected service time, we may further apply the approximation that cross traffic that entered after the update is still being serviced at t. Assuming the previous update was broadcast at time 0, the probability that there are no cross arrivals on all h links along a path between the previous update and time t is $e^{-\lambda_c th}$. Thus, the probability of a path being taken by cross traffic since the last update is:

$$P_{C|k}(t) \approx 1 - e^{-\lambda_c th} \tag{3.1}$$

Given these assumptions, the blocking probability, $\beta(t)$ can be expressed as:

$$\beta(t) = \sum_{k=0}^{K_k} P(K_o(t) = k) P_{C|k}(t)^k + \sum_{k=K_k+1}^{K_s} P(K_o(t) = k) P_{C|k}(t)^{K_k} \tag{3.2}$$

To compute the distribution of $K_o(t)$, we must first compute the steady state distribution of K_b (the number of paths available at each update). Note, in

the following computation, we have assumed the distribution of K_b is the same at each update. This implies the distribution of K_b at a particular update is independent of the value of K_b at the previous update. Although this assumption is not entirely accurate for small update intervals, it greatly simplifies the mathematics. We define $P(j)$ to be the probability that j paths are occupied with through traffic, $P(i)$ to be the probability that i paths are occupied by cross traffic and $P_{K_b}(k_b)$ as the steady state distribution of K_b. Since j denotes the number of paths taken by through traffic, and i denotes the number of paths taken by cross traffic, $K_s = K_b + i + j$. Summing over j, $P_{K_b}(k_b)$ can be written as:

$$P_{K_b}(k_b) = \sum_{j=0}^{K_s - k_b} P(j)P(i = K_s - j - k_b)$$

Using the approximation that all through arrivals are carried, we can model the entire system as two subsystems: one with through traffic only, and one with cross traffic only. The first system with through traffic is modeled as a $M/M/m/m$ queueing system where $m = K_s$. For an $M/M/K_s/K_s$ queueing system, the steady state probability that j paths are taken by through traffic is

$$P(j) \approx \frac{\rho_t^j / j!}{\sum_{n=0}^{K_s} \rho_t^n / n!}$$

In a similar manner, due to independence of cross traffic on all links, we can model the cross traffic on each link as a M/M/1/1 system. Each link is a single server queueing system that drops an arrival if there is there is already one arrival in the system. If the offered load is $\rho_c = \lambda_c / \mu$, the blocking probability for the link is $p = \frac{\rho_c}{1+\rho_c}$, and the probability a path is occupied is $1 - (1-p)^h$. The probability that i paths are taken by cross traffic, given that j paths are taken by through traffic is:

$$P(i) \approx (1 - (1-p)^h)^{K_s - j - K_b}$$

$P_{K_b}(k_b)$ can be written as a product of $P(i)$ and $P(j)$ summed over all possible j:

$$P_{K_b}(k_b) \approx \sum_{j=0}^{K_s-k_b} \left((1-(1-p)^h)^{K_s-j-k_b}\right) \times \frac{\rho_t^j/j!}{\sum_{n=0}^{K_s} \rho_t^n/n!} \tag{3.3}$$

Note that in computing the steady state distribution of K_b, there was an inherent assumption of independence between cross and through transactions. This assumption is only valid for a very small blocking probability. With this approximation, more through transactions are actually carried, thereby amplifying the blocking of newer through traffic arrivals by through traffic transactions in progress. For this reason, this model is only an approximation of the actual algorithm. However, because this analysis is geared for applications with low blocking, the approximation is very good.

Our objective is to design an algorithm to determine the number of paths that should be probed to achieve a given blocking probability at time t seconds after the last network state update. To that end, we need to know the distributions of the potentially open paths as time t after update evolves: $K_o(t) = K_b - K_T(t)$, the distribution of $K_o(t)$ may be expressed as:

$$P(K_o = k) = \begin{cases} \sum_{i=0}^{K_s} P(K_b = i) \times \left(1 - \sum_{l=0}^{i-1} P(K_T(t) = l)\right) & \text{if } k = 0 \\ \sum_{i=k}^{K_s} P(K_b = i) P(K_T(t) = i - k) & \text{if } k > 0 \end{cases} \tag{3.4}$$

Substituting the distributions of K_b and $K_T(t)$, the expression for $P(K_o(t) = k)$ is as follows:

$$P(K_o = k) = \begin{cases} \sum_{i=0}^{K_s} \sum_{j=0}^{K_s - i} \left((1 - (1-p)^h)^{K_s - j - i}\right) \times \frac{\rho_t^j / j!}{\sum_{n=0}^{K_s} \rho_t^n / n!} \times \\ \qquad \left(1 - \sum_{l=0}^{i-1} \frac{(\lambda_t t)^l e^{-\lambda_t t}}{(l)!}\right) & \text{if } k = 0 \\ \sum_{i=k}^{K_s} \sum_{j=0}^{K_s - i} \left((1 - (1-p)^h)^{K_s - j - i}\right) \times \frac{\rho_t^j / j!}{\sum_{n=0}^{K_s} \rho_t^n / n!} \times \\ \qquad \frac{(\lambda_t t)^{i-k} e^{-\lambda_t t}}{(i-k)!} & \text{if } k > 0 \end{cases} \tag{3.5}$$

Figures 3-4, 3-5, 3-5 and 3-6 illustrate the blocking probability as a function of time for various sets of K_s and λ_t. Figures 3-7, 3-8, 3-9, 3-10 show both $K_o(t)$ over time, as well as the values of K_k needed to maintain blocking probabilities less than or equal to 10^{-2}. Figures 3-11, 3-12, 3-13 and 3-14 combine Figures 3-4 - 3-6 to gain further insight.

With this algorithm, the information right after the network state update is accurate and yields the lowest blocking probability, whereas, towards the end of the interval between updates, new cross traffic may have joined the network

which leads to a higher blocking probability. Figure 3-4 shows that the update interval should be no longer than ~ 0.25 of the expected transaction service time (for 5 paths between source-destination) if the update information is of significant value. Figure 3-5 and 3-6 give update intervals of up to the expected transaction service time for a larger value of K_s. Figure 3-7 shows the number of paths required for Poisson traffic and exponential service time models and suggests that, unless the update interval is of the order of a quarter of the flow size, we must probe paths beyond our threshold value. Figure 3-9, where we have double the number of paths between source-destination, shows that the update interval may be increased to approximately the mean service time. From these results, we infer that if we only want to probe a few paths, we must constrict our update interval to a quarter of the mean service time. Otherwise, supposing we have the flexibility to probe a larger number of paths, a small blocking probability can still be maintained with longer update intervals. Although we have not provided results of the relationship between the blocking probability and expected arrival time, the update interval is equally a function of $1/\lambda_c$. Therefore while choosing an appropriate τ value, we must also be mindful of the arrival rate. This is synergistic with our desire to keep global network coordination speed slow and use quasi-static provisioning in the WAN for scalability of the network management system.

Although we conjecture that for high-performance, the optimal value of τ will be upper bounded by $1/\mu$, we must prove this fact by generalizing our computations in case 1 for all values of τ. The following section improves upon our approximation of the blocking probability for general τ.

Case 2: General τ

The expressions for the blocking probability in Case 1 had embedded in them two significant assumptions. The first, that through transactions had very small

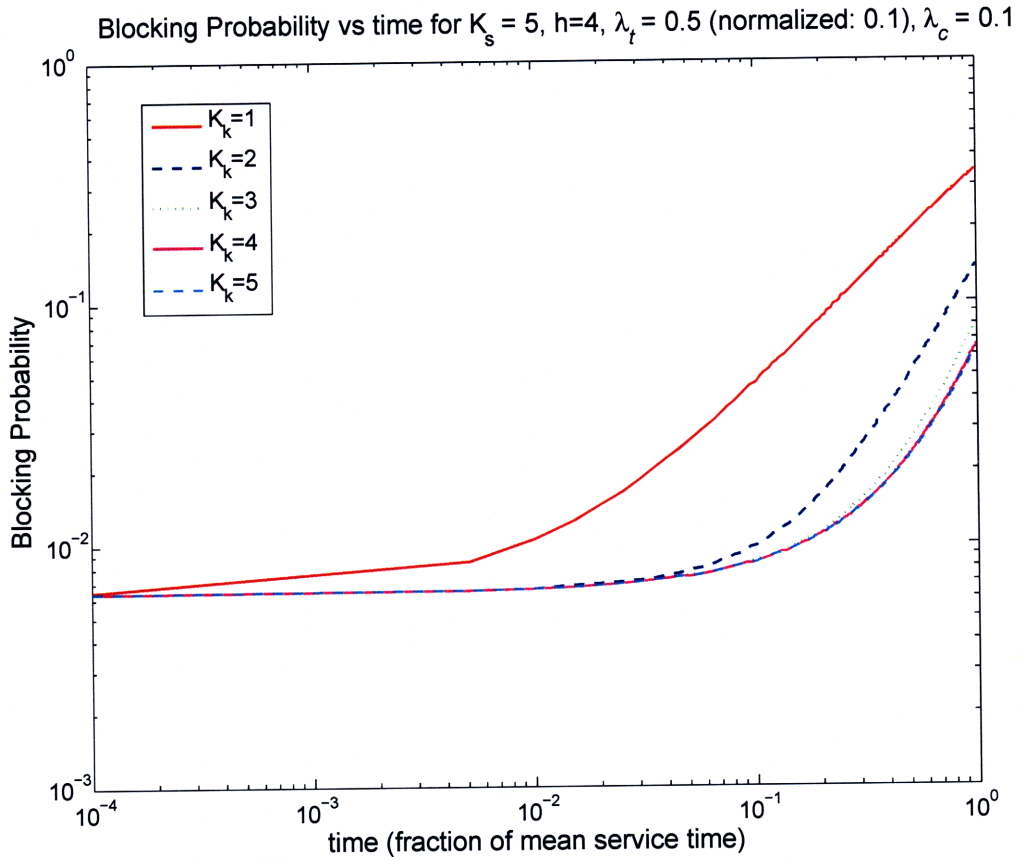

Figure 3-3: Blocking probability vs. time for various K_k. ($\lambda_t = 0.5$, $\lambda_c = 0.1$, $h = 4$, $K_s = 5$)

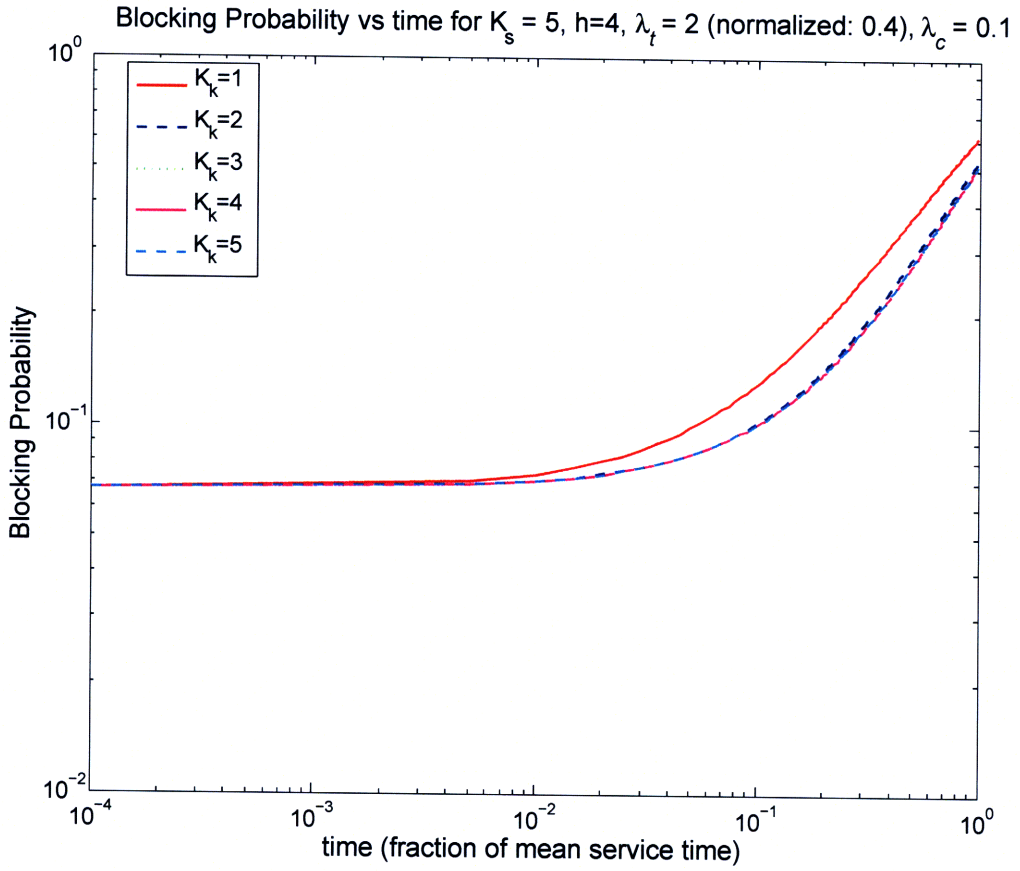

Figure 3-4: Blocking probability vs. time for various K_k.($\lambda_t = 2$, $\lambda_c = 0.1$, $h = 4$, $K_s = 5$)

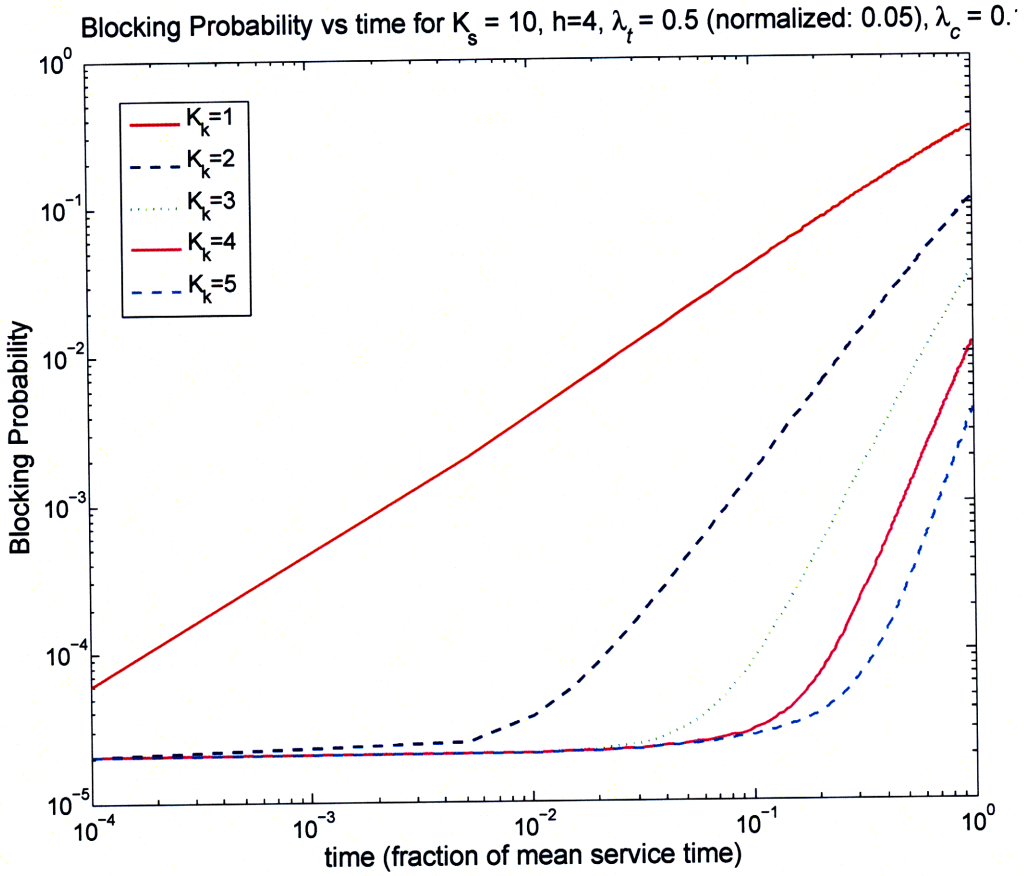

Figure 3-5: Blocking probability vs. time for various K_k.($\lambda_t = 0.5$, $\lambda_c = 0.1$, $h = 4$, $K_s = 10$)

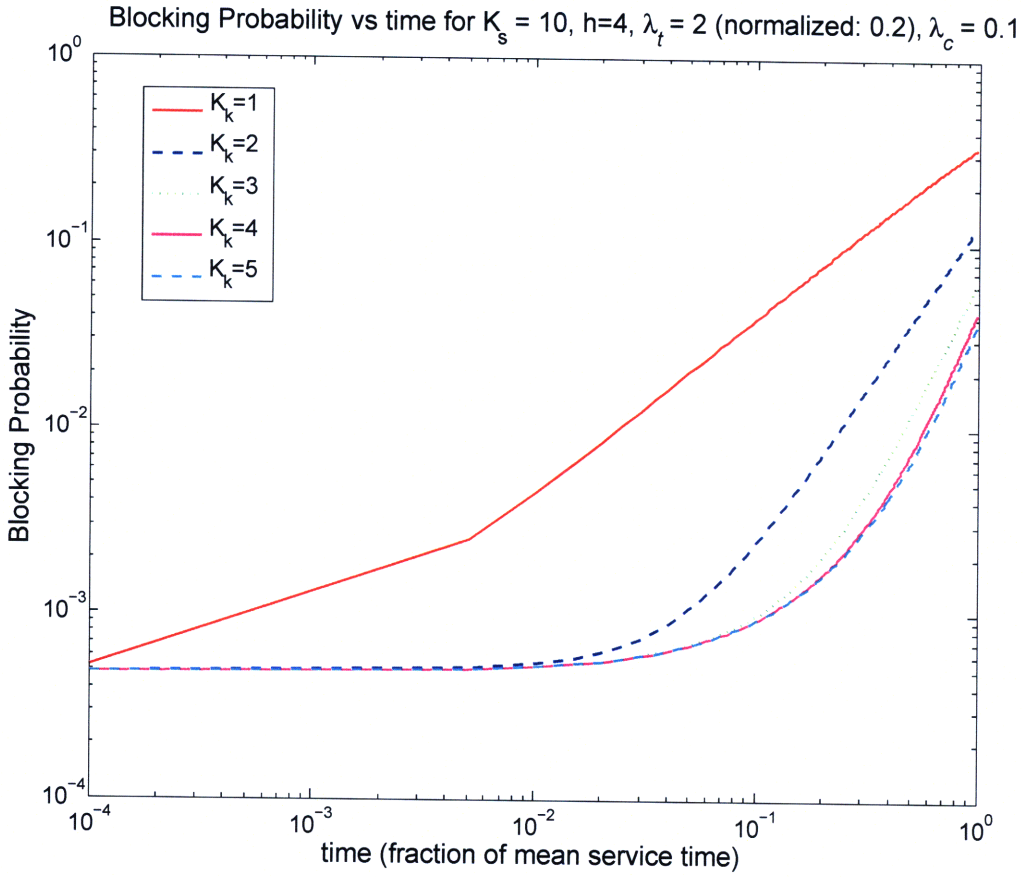

Figure 3-6: Blocking probability vs. time for various K_k.($\lambda_t = 2$, $\lambda_c = 0.1$, $h = 4$, $K_s = 10$)

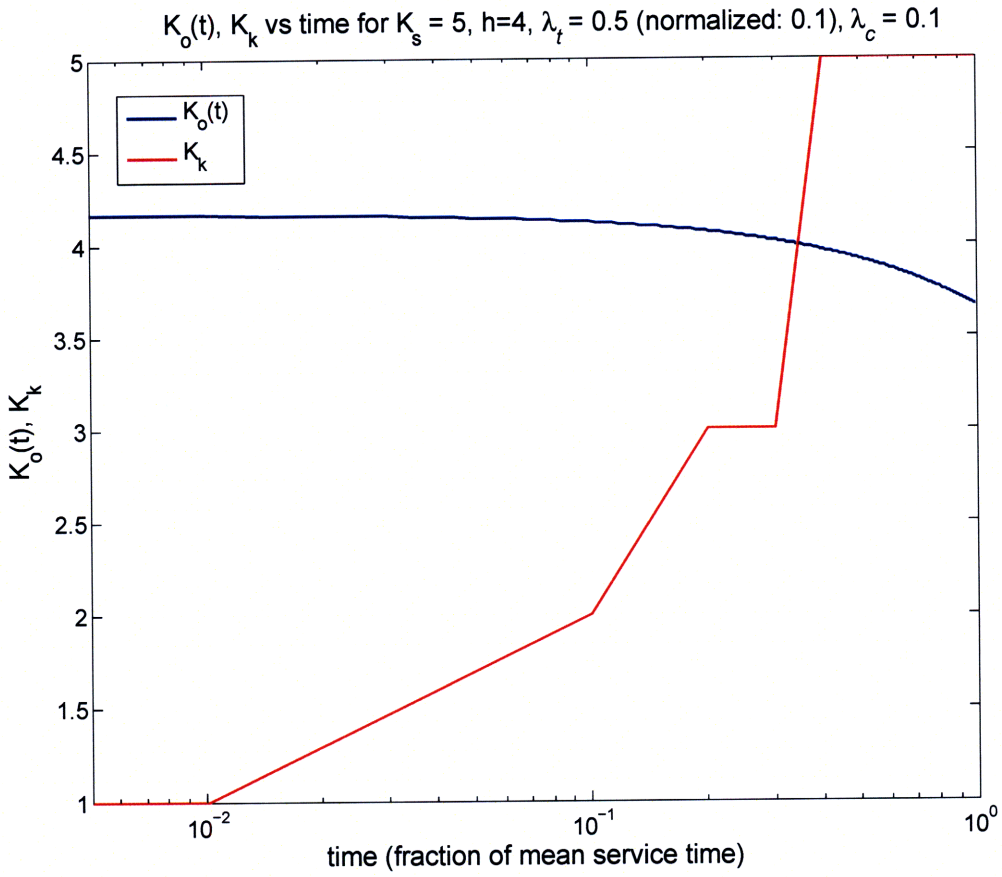

Figure 3-7: $K_o(t)$ and K_k vs. time. ($\lambda_t = 0.5$, $\lambda_c = 0.1$, $h = 4$, $K_s = 5$)

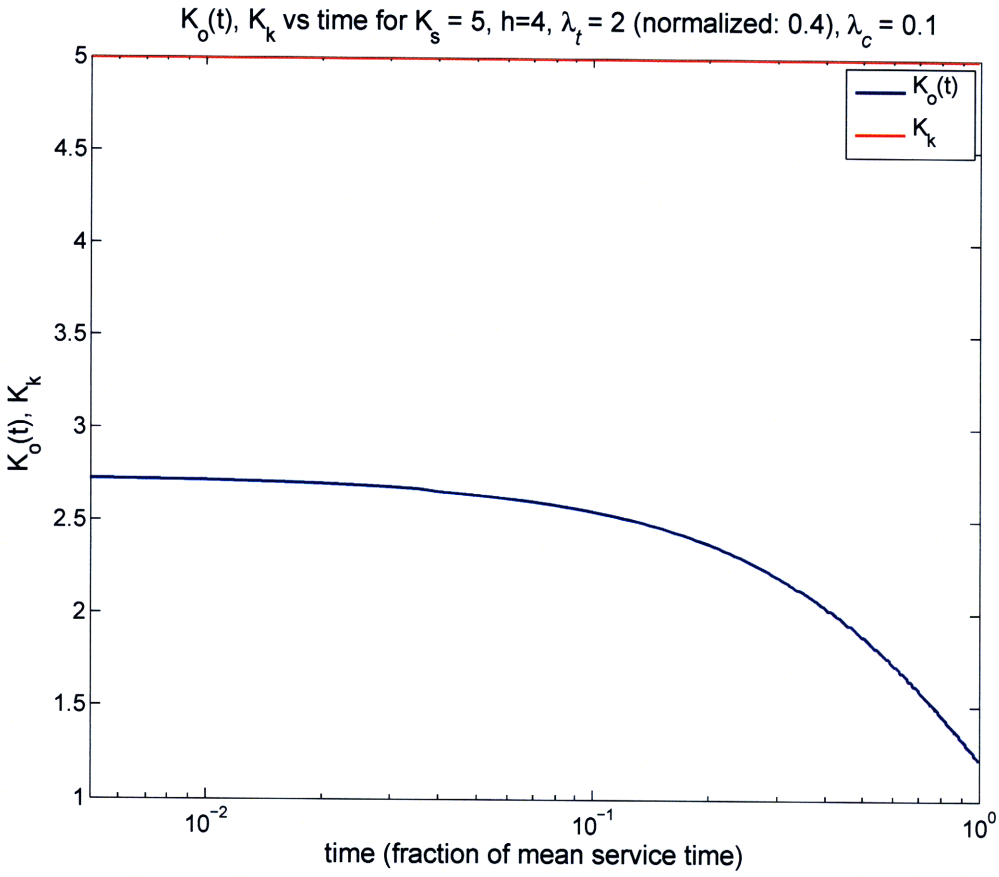

Figure 3-8: $K_o(t)$ and K_k vs. time. ($\lambda_t = 2$, $\lambda_c = 0.1$, $h = 4$, $K_s = 5$)

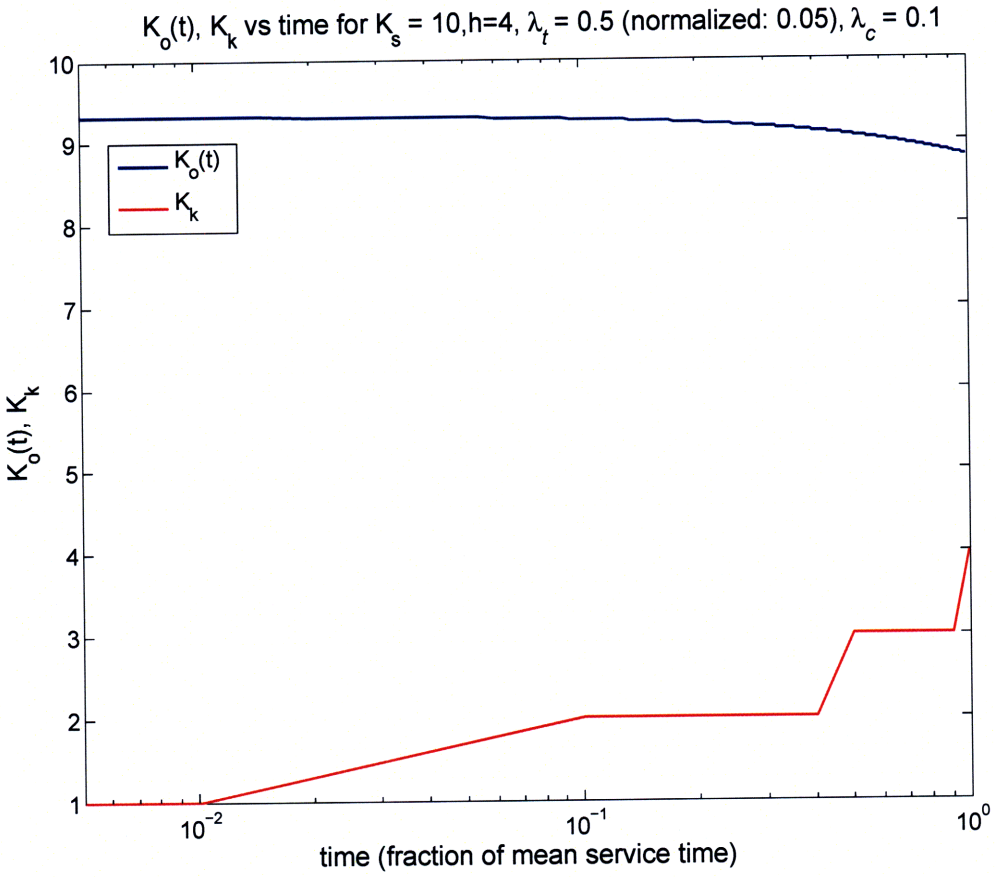

Figure 3-9: $K_o(t)$ and K_k vs. time. ($\lambda_t = 0.5$, $\lambda_c = 0.1$, $K_s = 10$)

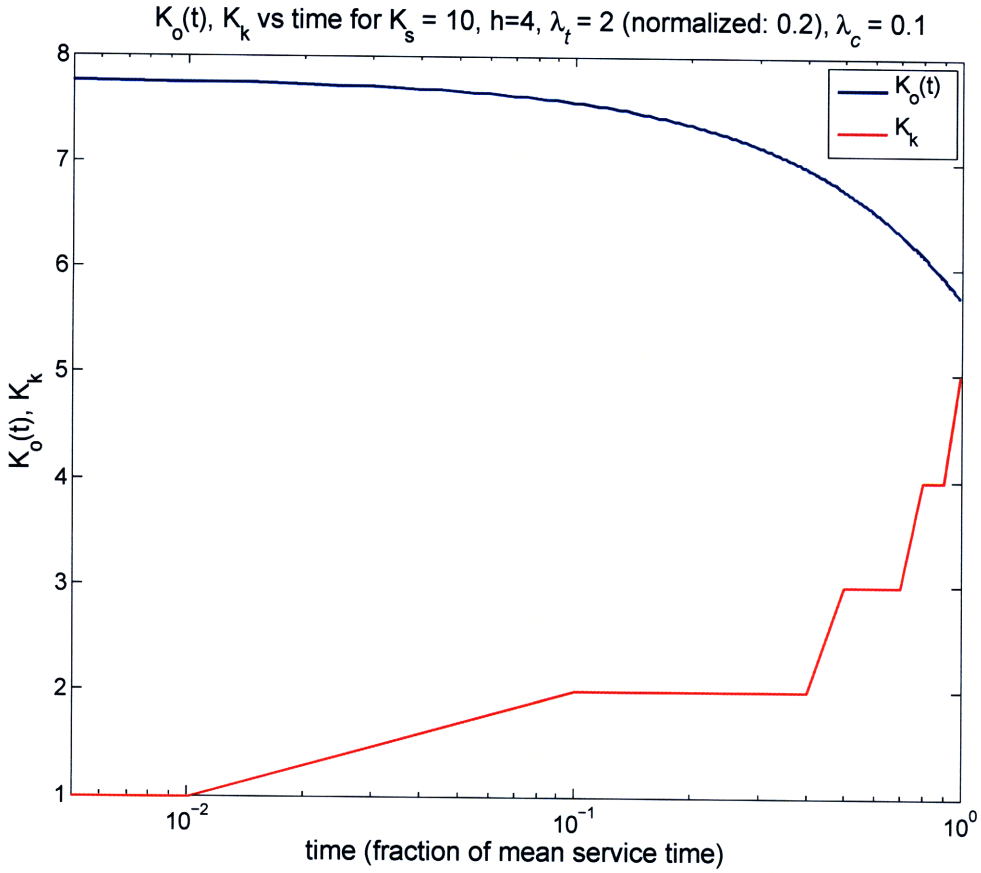

Figure 3-10: This is a plot of $K_o(t)$ and K_k vs. time. ($\lambda_t = 2$, $\lambda_c = 0.1$, $h = 4$, $K_s = 10$)

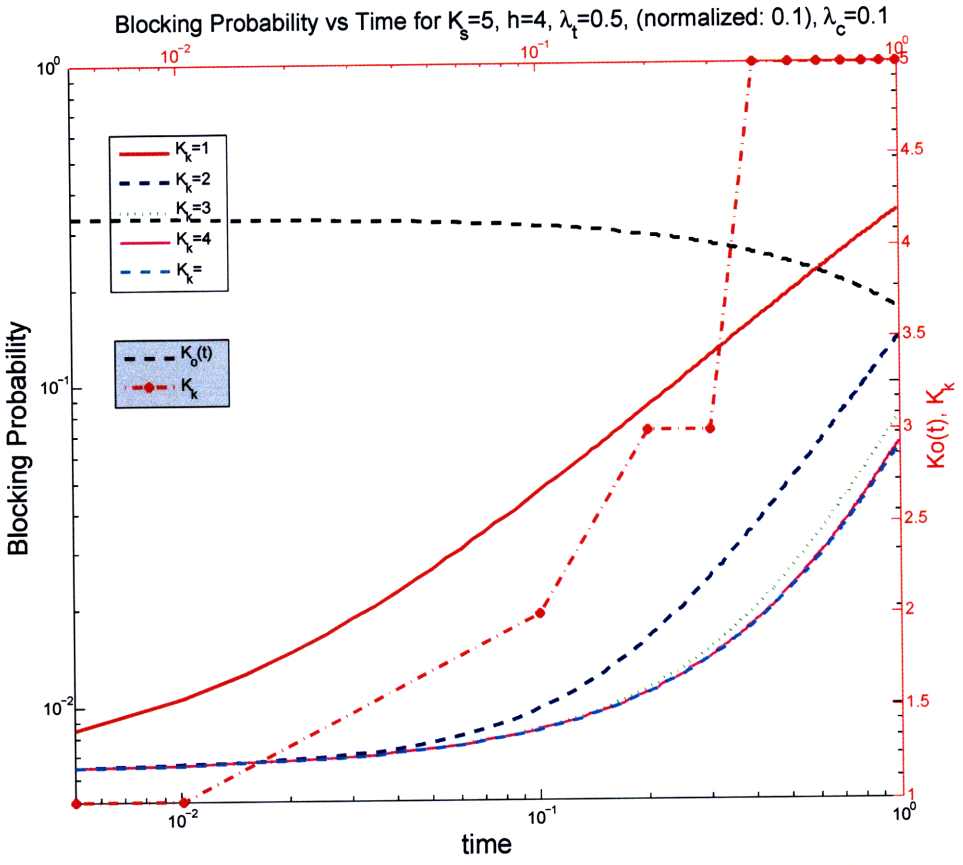

Figure 3-11: Combined plot of Blocking Probability, $K_o(t)$, and K_k vs. time. $\lambda_t = 0.5$, $\lambda_c = 0.1$, $h = 4$, $K_s = 5$.

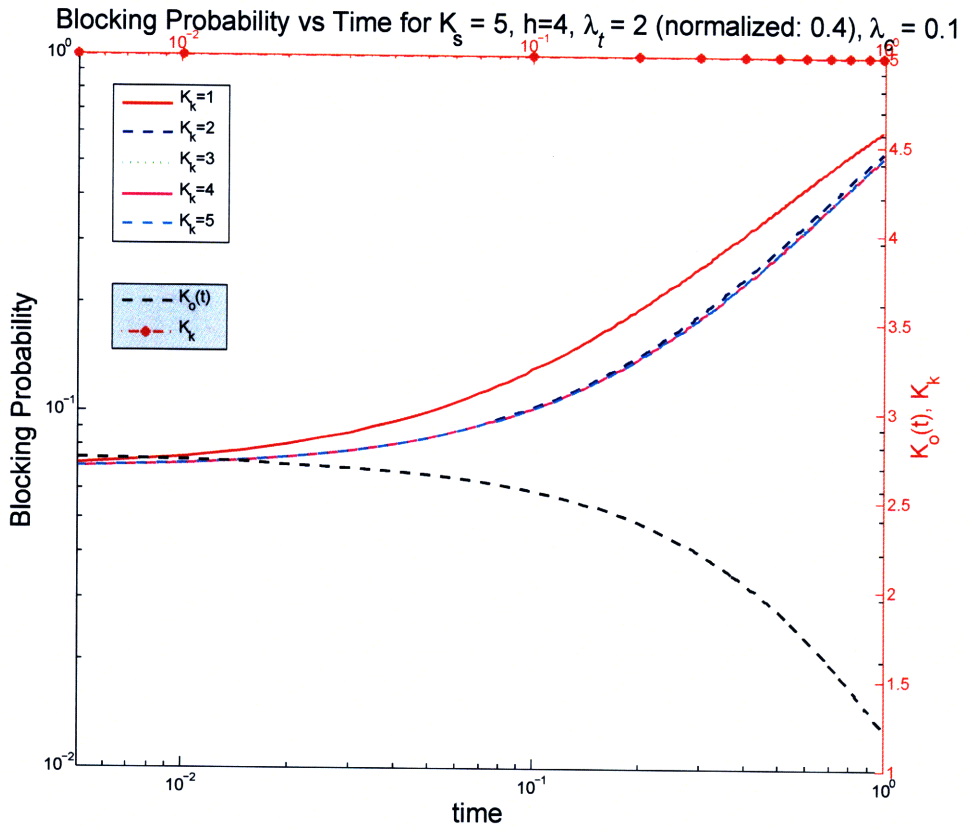

Figure 3-12: Combined plot of Blocking Probability, $K_o(t)$, and K_k vs. time. $\lambda_t = 2$, $\lambda_c = 0.1$, $h = 4$, $K_s = 5$.

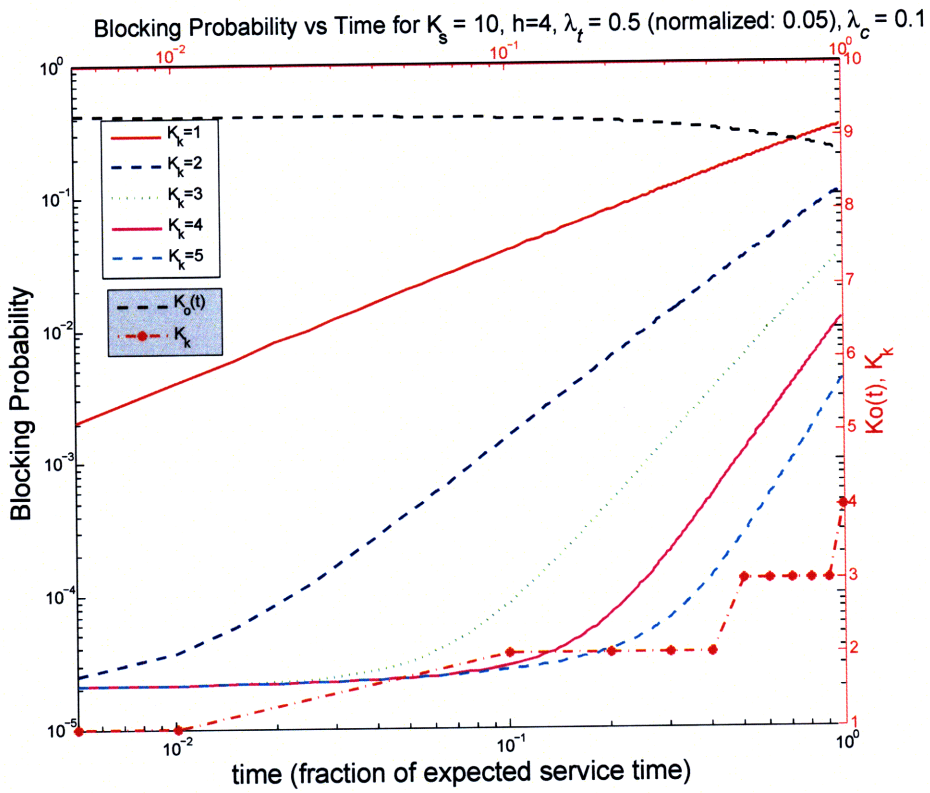

Figure 3-13: Combined plot of Blocking Probability, $K_o(t)$, and K_k vs. time. $\lambda_t = 0.5$, $\lambda_c = 0.1$, $h = 4$, $K_s = 10$.

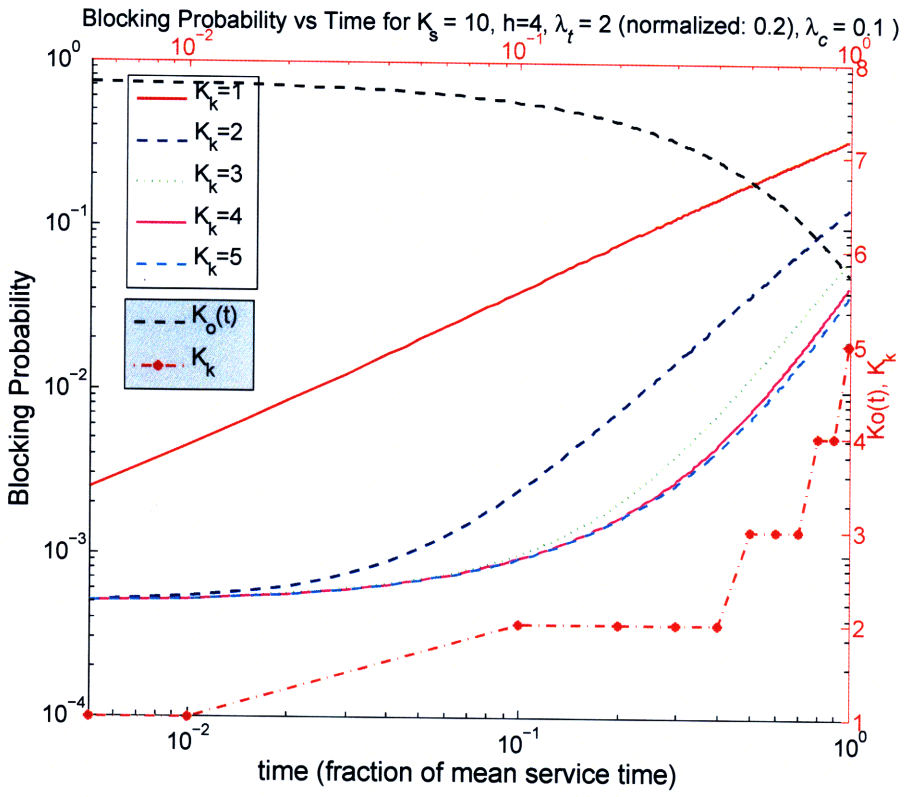

Figure 3-14: Combined plot of Blocking Probability, $K_o(t)$, and K_k vs. time. $\lambda_t = 2$, $\lambda_c = 0.1$, $h = 4$, $K_s = 10$.

blocking probability ($\leq 10^{-2}$), and second, that $\tau < min\{1/\mu, 1/\lambda_c\}$. The former of the two assumptions almost always holds as the contribution of this work is to design a scheduling approach for high performance applications. As far as the second assumption goes, our intuition tells us that the optimal value of τ will be upper bounded by $1/\mu$, but we must prove this fact by generalizing our computations in case 1 for all values of τ. The only two computations that depended on this restriction of τ were those for $P_{C|k}(t)$ and $P_{K_T(t)}(k_T(t))$. The general form of $P_{K_T(t)}(k_T(t))$ is very difficult to compute and outside of the scope of this thesis. Here we will generalize $P_{C|k}(t)$ to provide a better approximation to the previous case.

Cross traffic per link should not exceed a loading factor of $\approx 20\%$ since doing so could lead to significant blocking with higher hop-sizes. In addition, realistically τ is upper bounded by $\frac{2}{\mu}$, as beyond that, the state information is assumed to be stale and useless for high-performance operations. Abiding by these restrictions, [29] concluded that by limiting the number of cross arrivals per node to 2 significantly simplifies the transient analysis of the system, while offering a lower bound on the blocking probability. Given our assumption of a maximum of 2 arrivals per link since the last update, we may now compute $P_{C|k}(t)$. Since $P_{C|k}(t)$ is the probability that one of the paths in the subset $K_o(t)$ has been taken at time t, we must first compute the probability that a single link is occupied. Let us define the following variables:

X_1: Inter-arrival time of first arrival

X_2: Inter-arrival time of second arrival

Y_1: Duration of first arrival

Y_2: Duration of second arrival

X_1, X_2, Y_1, Y_2 all have exponential distributions with rates $\lambda_c, \lambda_c, \mu, \mu$ respectively. A link will be clearly blocked by either the first or second arrival. The

subsequent results are taken from [29]. There is blocking due to the following two events (1), and (2). (1) is the event where blocking is due to the first arrival, and (2) is the event where blocking is due to the second arrival. The probability a single link is blocked is $p = P(1) + P(2)$, and $P_{C|k}(t) = 1 - (1-p)^h$ as before.

$$(1)\ ((X_1 < t) \cap (X_1 + Y_1 > t))$$
$$(2)\ ((X_1 + Y_1 < t) \cap (Y_1 < X_2) \cap (X_1 + X_2 < t) \cap (X_1 + X_2 + Y_2 > t))$$

The probability of event (1) is:

$$\int_0^t \int_{t-x_1}^{\infty} p(x_1, y_1)\, dy_1\, dx_1.$$

The probability of event (2) is:

$$\int_0^{\infty} \int_0^{t-y_1} \int_0^{\infty} \int_{max(y_1, t-y_2-x_1)}^{t-x_1} p(x_1, x_2, y_1, y_2)\, dx_2\, dy_2\, dx_1\, dy_1.$$

The expression that results from the integration in (1) is

$$\frac{\lambda_c e^{-\mu t}}{\mu - \lambda_c} \left(e^{t(\mu - \lambda_c)} - 1\right)$$

The expression that results from the integration in (2) is

$$\frac{\mu}{\lambda_c}(-3e^{-(\lambda_c+\mu)t} - \lambda_c t e^{-(\lambda_c+\mu)t} + 3e^{-\mu t} + \frac{1}{2}\lambda_c^2 t^2 e^{-\mu t} - 2\lambda_c t e^{\mu t} + 2e^{-(\lambda_c+\mu)t} + \lambda_c t e^{-(\lambda_c+\mu)t} - 2e^{-\mu t} + \lambda_c t e^{\mu t})$$

Figures 3-15, 3-16, 3-17 and 3-18 illustrate the new blocking probability curves for case 2 in comparison to those of case 1. These figures show that there is little difference between the approximation in case 1 and 2 for large update intervals. This is mainly due to the fact that now we assume that cross traffic may arrive and complete the transaction before time t. Since cross arrivals may leave the system, the

probability of occupancy for each link slightly decreases for longer update intervals ($\sim$ expected service time). For the performance thresholds that we are interested in however ($P_B \leq 10^{-2}$), the approximations for case 1 and 2 give the same blocking probability curves, yielding the same update intervals and K_k as those calculated in case 1.

It can be argued that for high-performance applications with bursty, large transactions, allowing pre-emptive power for users is more efficient than engineering a fast acting centralized scheduling system which requires immense sensing/reporting and processing capabilities and is not scalable. However, pre-emption is very disruptive even for second-class traffic. Thus, we envision flow switching would be used in cases of large transactions ($>$ 1 s) with network updates that can be slow provided that fast probing of multiple paths is used in combination.

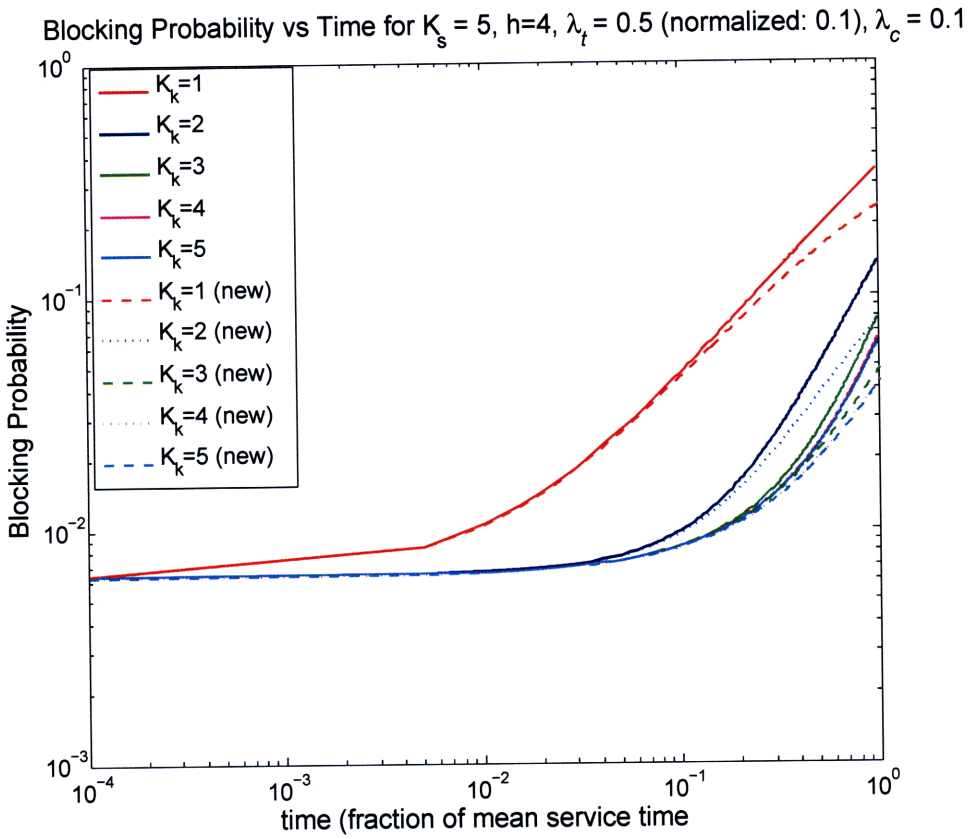

Figure 3-15: This is a plot of blocking probability vs. time. $\lambda_t = 0.5$, $\lambda_c = 0.1$, $K_s = 5$

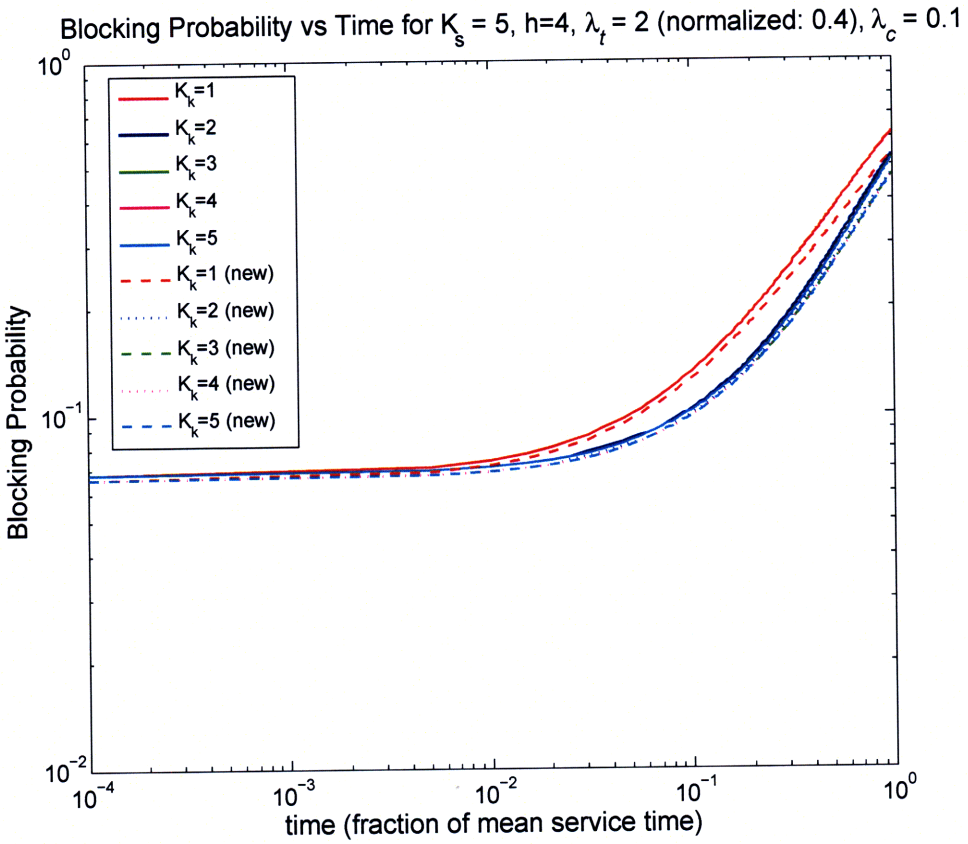

Figure 3-16: This is a plot of blocking probability vs. time. $\lambda_t = 2$, $\lambda_c = 0.1$, $K_s = 5$

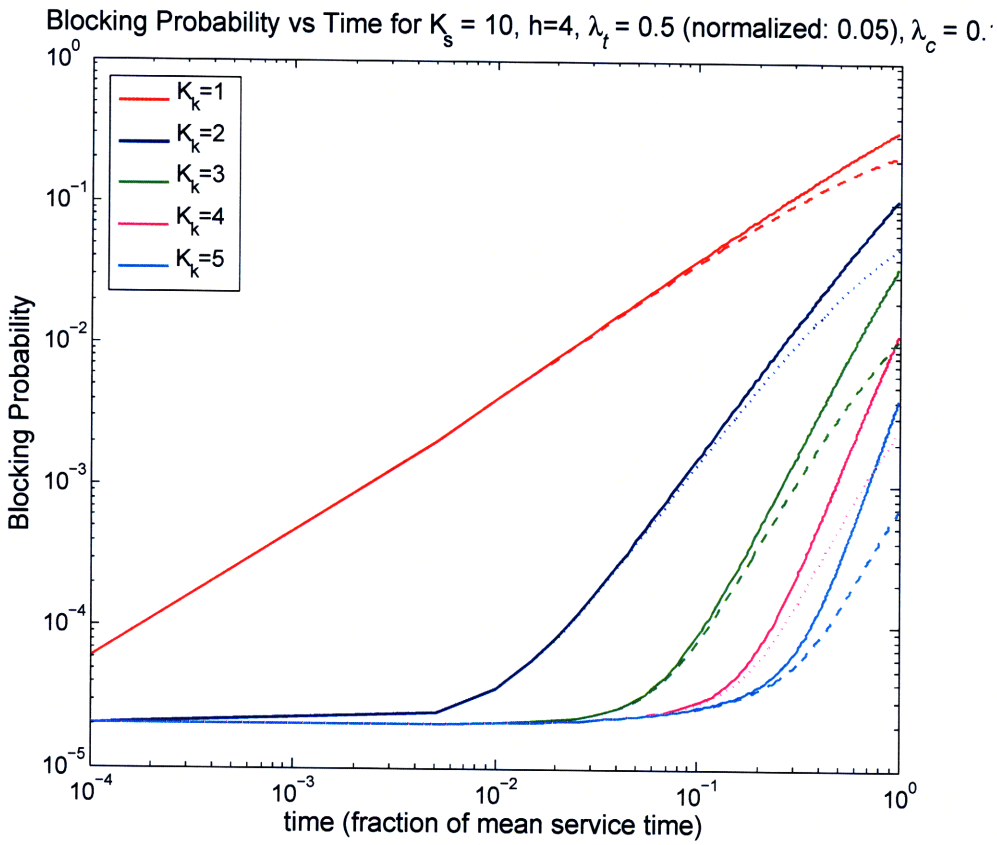

Figure 3-17: This is a plot of blocking probability vs. time. $\lambda_t = 0.5$, $\lambda_c = 0.1$, $K_s = 10$

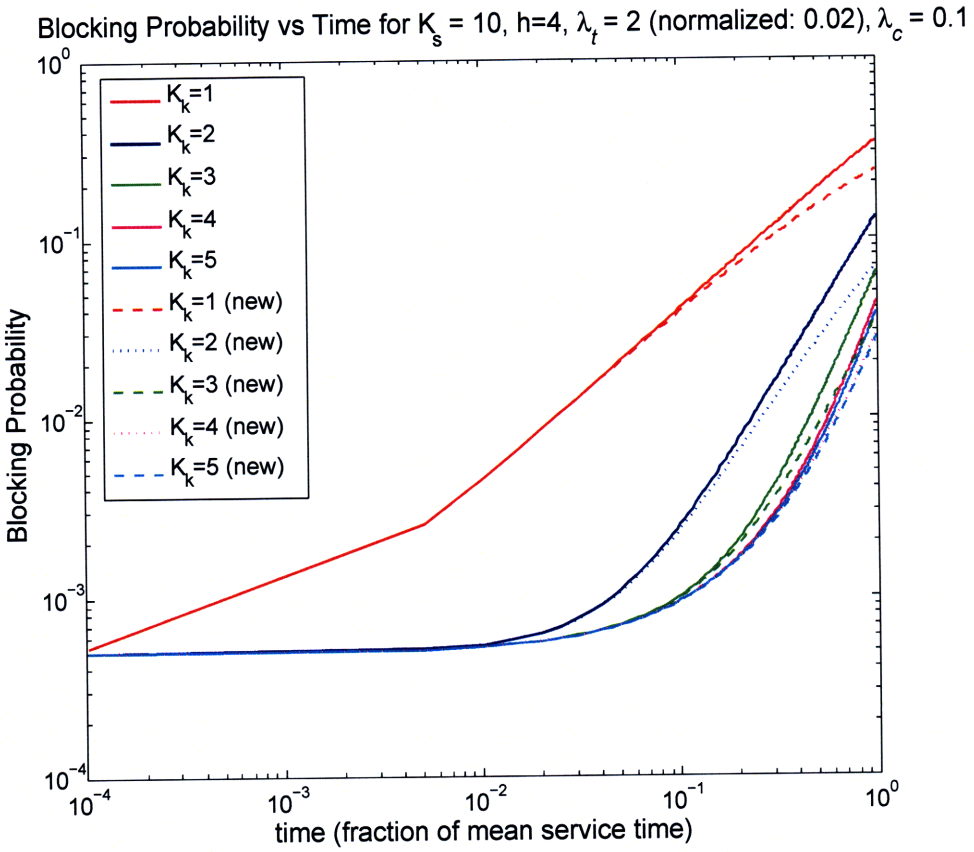

Figure 3-18: This is a plot of blocking probability vs. time. $\lambda_t = 2$, $\lambda_c = 0.1$, $K_s = 10$

Chapter 4

Results and Comparison of Candidate Scheduling Approaches

4.1 Overview of Results

The purpose of this thesis is to design an implementation scheme for online OFS network management and control of future high-performance networks serving applications with stringent delay requirements. To conclude our comparative study, we first summarize the performance metrics for each of the scheduling implementations. The following results are illustrated for a blocking probability of 10^{-2} for through traffic.

Algorithm 1: (Preemptive power for through traffic)
For this algorithm, which gives pre-emptive power to high-performance users - the total offered through traffic λ_t is upper bounded by 1.4 (normalized load per path ≈ 0.3 for 5 paths). This scheme maximizes throughput for through traffic, but at a severe cost as the performance of cross traffic may be significantly affected.

Algorithm 2a: (No preemptive power for through traffic & no correlation between links)
For hop-size $h = 3$, and an upper bound of 5 on the maximum number of paths we can probe: p, the probability of link occupancy, cannot exceed 0.1

Algorithm 2b: (No preemptive power for through traffic with correlation between links)

Factoring in correlation between subsequent links yielded approximately the same maximum loading as algorithm 1 for hop-sizes up to 4 (correlation coefficient of 0.5). However, as shown in chapter 2, the model does not represent the algorithm accurately for high correlation. In any case, beyond $h = 4$, the performance of even this model degrades in that the values of K_s for the same offered load in algorithm 1 are significantly less than the corresponding values for this algorithm.

Algorithm 3:

We used slightly different nomenclature in this section: K_s is still the total number of paths, but the main parameter of interest that we have constrained over is $K_p(t)$ (the number of paths we must probe). For the following cases, the update intervals are reported, by again ensuring $K_p(t) \leq 5$, and $P_B = 10^{-2}$.

- $K_s = 5, \lambda_t = 0.5$(normalized 0.1), $\lambda_c = 0.1, h = 4$: $\tau \approx 1/4\mu$ sec
- $K_s = 5, \lambda_t = 2$(normalized 0.4), $\lambda_c = 0.1, h = 4$: There are no values of τ that ensure our performance thresholds.
- $K_s = 10, \lambda_t = 0.5$(normalized 0.05), $\lambda_c = 0.1, h = 4$: $\tau \approx 1/\mu$ sec
- $K_s = 10, \lambda_t = 2$(normalized 0.2), $\lambda_c = 0.1, h = 4$: $\tau \approx 1/2\mu$ sec

4.1.1 Comparative Analysis

Algorithm 1 requires pre-emption and is too disruptive to be appealing, but it sets a lower bound on the number of required paths between source-destination pairs at a given network loading. It indicates that the price to pay for stringent time deadline service is over-provisioning of the network and light traffic loading, [29]. In [6,28] we indicated how to get back to high utilization by using dedicated lightpaths in the WAN between MANs and a Media Access Control (MAC) Protocol for statistical multiplexing. However, that technique requires

scheduling and would not meet the stringent time deadline requirement of approximately one roundtrip delay. Algorithm 2 can be used in a network with considerable loading and achieve low blocking probability by probing all possible paths between the source-destination node pair. For large diameter networks, however, this can entail the use of a large amount of network resources.

With algorithm 3 we demonstrated a plausible integrated scheduling scheme, which uses a global control plane to probe candidate paths. The most frequent update rate, τ, was found for the case where $K_s = 5$, $\lambda_t = 0.5$ (normalized load per path: 0.1). Updating the entire network state this frequently overload the control plane, therefore we must determine the overall feasibility of the worst-case scenario. There are three types of delays incurred that are of importance: The computation time to recompute network state at the scheduler (every τ seconds), which must then broadcast update packets to all nodes in the network. At the user node, there are processing delays of the updates, which are used to choose a set of paths to probe. Lastly, there is the delay associated with the time it takes to probe the paths. As a result, an unmanageable update rate could lead to unwanted scheduling and processing delays for the user. The two main constraints for the update interval are: First, it must be larger than the summation of the average round-trip propagation delay, computation time for network state reconfiguration at the scheduler, and processing delay of updates at user nodes. Second, periodic broadcasts of network state information must not exceed data rate limitations of the control channel.

An estimate of the round-trip propagation delay can be found by computing the time it takes a light-path to traverse across the US. For instance, take a source-destination pair in Los Angeles and Boston respectively. Let's say that they are separated at a distance of approximately 3000 miles (about 4828 km), which is a roundtrip distance of roughly 9656 km. Assuming a worst-case velocity of an optical signal of 1.5×10^8, the time to travel from LA to Boston and back is $9.6 \times 10^6 / 1.5 \times 10^8 \approx 64$ milliseconds. This serves as a very optimistic lower

bound on τ since there is still the added delay of processing at user nodes and computation at the scheduler. For $\tau = 0.25$ secs, this added delay factor must be less than 0.2 secs, which provides a very small margin. For a larger number of independent paths, ($K_s = 10$) update intervals are much more reasonable (0.8 sec and 1 sec). This case provides a much greater margin for delays at the user and scheduler, and therefore serves as a more feasible design. However you have the added risk that there may or may not be 10 independent paths across the backbone for all source-destination pairs given average node degrees of 3.

As far as control channel limitations are concerned, let us look at the following calculation. Consider the scenario where the scheduler must report on 100 nodes in the backbone at each update. Since the average node degree is 3, we can assume there are roughly 150 links. Let us assume each link has 100 wavelengths, there are $15,000$ wavelength channels. If each wavelength channel requires 1.5 Kbyte (one IP packet) of information on average at each update, then the average control transaction size is upper bounded by 1 Gigabyte. In the worst case instance where $\tau = 0.25$ secs, we get an update date rate of 1 Gbyte/0.25 secs $\approx$ 4 Gbytes/sec (32 Gbps). A control channel of this nature can support 10 Gbps/sec links. We would then need three 10 Gbps channels or one 40 Gbps channel.

For large networks, Algorithm 3 makes use of periodic network broadcasts (a slow process) in conjunction with probing a small number of paths (a fast process) to achieve the same low blocking probabilities with high network loading. We believe that this is the most scalable network architecture for ultra-high-performance OFS networks. To increase network loading without compromising the performance of this fast service, pre-emptable scheduled flows could be simultaneously supported by the network, [29] for even better cost efficiencies. [30]

Chapter 5

Bibliography

[1] Greg Bernstein and Bala Rajagopalan, Introduction to optical control plane standards and technology: OIF UNI, GMPLS, G.ASON and all that, (Optical Fiber Communication Short Course Notes), March 18, 2002

[2] Bishwaroop Ganguly and Eytan Modiano, Distributed Algorithms and Architectures for Optical Flow Switching in WDM networks, in (Proceedings of ISCC 2000), pp 134-139.

[3] J. Li et al. Dynamic routing with inaccurate link state information in integrated IP-over-WDM networks, Computer Networks, Volume 46, Issue6, December, 2004

[4] S Shen et al. Benefits of advertising wavelength availability in distributed lightpath establishment, Computer Networks, Volume 50, Issue 13, September, 2006

[5] Dimitri Bertsekas and Robert Gallager, Data Networks, second edition, 1992, Prentice-Hall

[6] Guy Weichenberg, "Design and Analysis of Optical Flow Switched Networks": Ph.D. Dissertation, Massachusetts Institute of Technology, 2008

[7] Mark E Crovella and Murad S. Taqqu and Azer Bestavros, Heavy-tailed probability distributions in the World Wide Web, (A Practical Guide to Heavy Tails:Statistical Techniques and Applications), Birkhauser, Boston(1998), pp 3-25

[8] Xiaoyun Zhu and Jie Yu and John Doyle Heavy Tails, Generalized Coding, and Optimal Web Layout, (Proceedings of IEEE INFOCOM 2001), pp 1617-1626

[9] Guy Weichenberg, "High-Reliability Architectures for Networks under Stress", Masters Thesis, Massachusetts Institute of Technology, 2003

[10] Guy Weichenberg, "Design and Analysis of Optical Flow Switched Networks", PhD Thesis Defense, November 6, 2008

[11] V. W. S. Chan, K. L. Hall, E. Modiano, and K. A. Rauschenbach, Architectures and technologies for high-speed optical data networks, IEEE/OSA Journal of Lightwave Technology, vol. 16, no. 12, pp. 21462168, 1998.

[12] V. W. S. Chan, Editorial, IEEE Journal on Selected Areas in Communications: Optical Communications and Networking Series, vol. 23, no. 8.

[13] S. B. Alexander et al., A Precompetitive Consortium on Wide-Band All-Optical Networks, IEEE/OSA Journal of Lightwave Technology, vol. 11, no. 5/6, pp. 714-735.

[14] All-Optical Networking Consortium Website: http://www.ll.mit.edu/aon/.

[15] V. W. S. Chan, K. L. Hall, E. Modiano, and K. A. Rauschenbach, Architectures and technologies for high-speed optical data networks, IEEE/OSA Journal of Lightwave Technology, 1998.

[16] V.W.S. Chan, Editorial, IEEE, JSAC Vol. 23 No. 8, August 2005.

[17] V.W.S. Chan, Editorial: Optical Network Architecture from the Point of View of the End User and Cost, IEEE Journal on Selected Areas in Communications, Optical Communications and Networking, Volume 24, Issue 12, pp. 1-2, December 2006.

[18] V. W. S. Chan, G. Weichenberg, M. Mdard, Optical Flow Switching, Workshop on Optical Burst Switching, San Jose, October 2006 (invited).

[19] V.W.S. Chan, Editorial: Hybrids, IEEE Journal on Selected Areas in Communications, Optical Communications and Networking, Volume 25, Issue 4, page 1, April 2007.

[20] V.W.S. Chan, Editorial: Free Space Optical Networking Over the Atmosphere, IEEE Journal on Selected Areas in Communications, Optical Communications and Networking, Volume 25, Issue 6, page 1, August 2007.

[21] V.W.S. Chan, Editorial: Near-Term Future of the Optical Network in Question?, IEEE Journal on Selected Areas in Communications, Optical Communications and Networking, Volume 25, Issue 9, page 1, December 2007.

[22] MIT Lincoln Labs, Advanced Networks Group NGI-ONRAMP Consortium Website: http://www.ll.mit.edu/AdvancedNetworks/ngi.html

[23] On the throughput-cost tradeoff of multi-tiered optical network architectures, G. Weichenberg, V. W. S. Chan, and M. Medard, Globecom 2006, San Francisco, November. 2006.

[24] G. Rossi, T.E. Dimmick and D. J. Blumenthal, Optical performance monitoring in reconfigurable WDM optical networks using subcarrier multiplexing, IEEE/OSA Journal of Lightwave Technology, Vol. 18, No. 12, December 2000, pp. 1639-1648.

[25] G. Weichenberg, V.W.S. Chan, M. Mdard, Cost-Efficient Optical Network Architectures, 32nd European Conference on Optical Communication (ECOC), Cannes, France, September 2006.

[26] G. Weichenberg, V.W.S. Chan, and M. Medard, On the Capacity of Optical Networks: A Framework for Comparing Different Transport Architectures, Optical Communications and Networking Supplement of the IEEE Journal on Selected Areas in Communications, Volume 25, Issue 6, pp. 84 101, August 2007

[27] G. Weichenberg and V.W.S. Chan, Access Network Design for Optical Flow Switching, Proceedings of IEEE Global Telecommunications Conference (Globecom 2007), Washington, D.C., Nov. 2007.

[28] G. Weichenberg, V.W.S. Chan, and M. Medard, Throughput-Cost Analysis of Optical Flow Switching, Optical Fiber Conference, San Diego, March 23, 2009.

[29] Bishwaroop Ganguly, Implementation and Modeling of a Scheduled Optical Flow Switching (OFS) Network, PhD Thesis EECS June 2008.

[30] V.W.S. Chan, Anurupa Ganguly, Guy Weichenberg, "Optical Flow Switching with Time Deadlines for High-Performance Applications"