

MIT Open Access Articles

*Optical flow switching with time deadlines
for high-performance applications*

The MIT Faculty has made this article openly available. **Please share**
how this access benefits you. Your story matters.

Citation: Chan, V.W.S., A.R. Ganguly, and G. Weichenberg. "Optical Flow Switching with Time Deadlines for High-Performance Applications." Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE. 2009. 1-8. ©2010 Institute of Electrical and Electronics Engineers.

As Published: <http://dx.doi.org/10.1109/GLOCOM.2009.5426119>

Publisher: Institute of Electrical and Electronics Engineers

Persistent URL: <http://hdl.handle.net/1721.1/58961>

Version: Final published version: final published article, as it appeared in a journal, conference proceedings, or other formally published context

Terms of Use: Article is made available in accordance with the publisher's policy and may be subject to US copyright law. Please refer to the publisher's site for terms of use.



Optical Flow Switching with Time Deadlines for High-Performance Applications

Vincent W. S. Chan

Claude E. Shannon Communication and
Network Group, RLE
Massachusetts Institute of Technology

Anurupa R. Ganguly

Claude E. Shannon Communication and
Network Group, RLE
Massachusetts Institute of Technology

Guy Weichenberg

Claude E. Shannon Communication and
Network Group, RLE
Massachusetts Institute of Technology

Abstract— This paper 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. This paper proposes 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.

Keywords: *Optical Network, Optical Flow Switching, Network management and Control, All-Optical Networks, Physical Layer Switching*

I. INTRODUCTION

In recent works, [6,10-19, 21-23,25-29] we 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 future increases in data volume per transaction (two to three orders of magnitude). 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 have stringent time deadline and would not like to wait in a queue but will be willing to pay for more expense to use the network. In this paper we explore a fast OFS for stringent time deadline services. This fast flow switching architecture is highly efficient and economical for on-demand high data rates 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 paper 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 paper:

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.

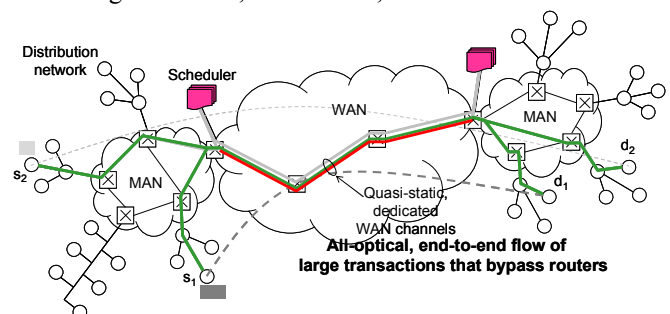


Figure 1. OFS with transparent, end-to-end data flow between users.

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 ($>100\text{ms}$ 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 2 shows the increase in utilization 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 paper 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.

Connection setup in the WAN has been studied previously in numerous works. In [1], a next-generation optical network architecture, Generalized Multiprotocol Label Switching (GMPLS) is presented. Although GMPLS can entail all-optical connection setup in the WAN as in OFS, it differs in the LAN and MAN design. In [1], the authors conclude that network reconfigurations on the order of minutes are required in order to maintain feasible control packet overhead. In [2,4], a scheduling algorithm is outlined in which the user reserves a lightpath based on network information that is periodically updated. Both works reported that these implementations reduced blocking probability for small update intervals. However, significant blocking may result in stale network information for reasonable control traffic and delay. In [3], a scheme is studied with updates that are based on triggered mechanisms rather than periodic ones. Here, the network state

is updated by nodes from time to time as a result of significant changes in link status, as observed by individual nodes. Owing to the nature of these updates, [3] showed that this approach also results in stale network information and incurs very high delays of updates. In this paper, we propose to probe multiple paths simultaneously and achieve a good target blocking probability even with stale network state information, or with no state information at all except for average loading of the network.

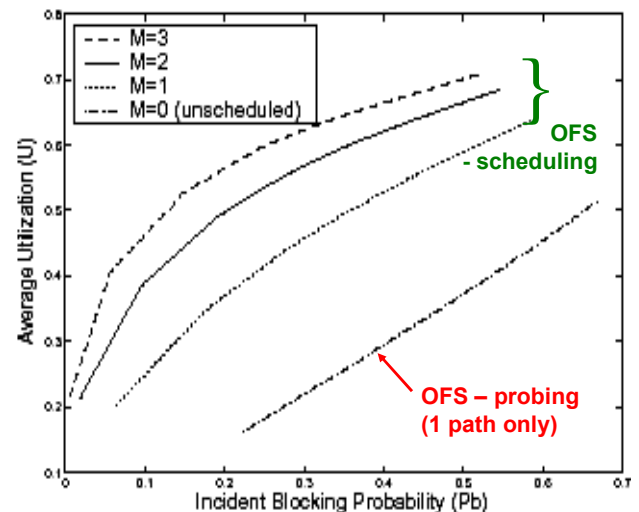


Figure 2. OFS with and without scheduling horizons, [29], showing increased utilization with increasing scheduling horizon (M transactions) for the same blocking probability performance.

II. OFS PROTOCOLS AND CONTROL ARCHITECTURE

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. Since wavelength converters are still too expensive for deployment, for the rest of the paper we have assumed no wavelength conversion and each path then is one specific wavelength in a fiber path. Different wavelengths on the same fiber path are considered different paths. The network topology relevant to this work is a general well connected mesh topology. At first glance it seems that the number of independent fiber paths is limited to the fiber degree of the source and destination nodes but in fact the source and destination nodes will be aware of the open and busy paths immediately going out or in from the nodes. What we are treating is actually the paths beyond the first hop. Though the examples use a target blocking

probability of 10^{-2} , clearly the algorithms and the techniques are valid for any desired blocking probabilities.

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 neither 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.

III. 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).

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.

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.

Algorithm 3: In this algorithm through traffic has no pre-emptive priority. Network state (link-states) information is periodically broadcast (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 wavelengths 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 (~ 5 ms).

IV. PERFORMANCE COMPARISON

In this section we analyze the performance of the three algorithms. Both Algorithms 1 and 2 are very restrictive and

*These paths are computed in an offline, centralized fashion and only have to change slowly based upon long-term traffic changes.

†Moreover, these aggressive requirements also dictate the need for active probing of unused network resources to ensure that they are functional, unless one is resigned to wasteful resource allocation for diversity routing in case of undetected failures.

‡In the event that a session has to be k -protected, it is expected that tying up $k-1$ lightpaths for 100 ms will not substantially waste resources (at most the average session duration divided by the setup time (< 100 ms)).

disruptive and should probably never be used. Their performance is used here to illustrate the attractive and unattractive features of particular types of architectures.

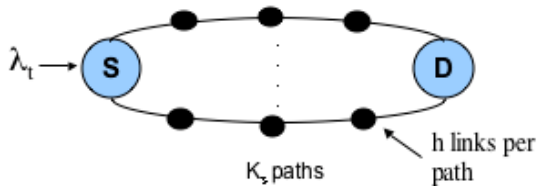


Figure 3. Algorithm 1: Through traffic pre-empts all other traffic internal to the paths except for those originating from the same source node. Traffic arrives as a Poisson arrival process at source node s destined for node d with K_s number of total paths, each with h links, available.

Algorithm 1: Figure 3 depicts source s and destination d connected by K_s total paths[§] each with h links^{**}. The through traffic at node s is modeled as a single Poisson process of rate λ_t and exponentially distributed transactions 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 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. The blocking probability is given by the Erlang B formula for an $M/M/k/k$ queue ($k=K_s$):

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

Algorithm 2: In this approach, through traffic from s to d as well as cross traffic at the links of each path are both present. To gain some analytical insights, we simplified the statistical model to assume the traffic arriving at each link is independent of one another with p being the probability that a link is serving a transaction. An arrival from s to d will be blocked if at least one link of each path is being used on all K_s paths. Since 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 after the last broadcast update. 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 blocking probability for this system is:

[§] We had assumed the paths that are candidates are link disjoint in that no two paths share any common links.

^{**} This can be easily generalized to a different number of links for each path.

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

For a network with heavy loading, Algorithm 2 will have to probe a substantial fraction of all, if not all, paths connecting s and d . Algorithm 3, discussed next, is a hybrid system with periodic network state updates to significantly reduce probing efforts.

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, 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. 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 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.

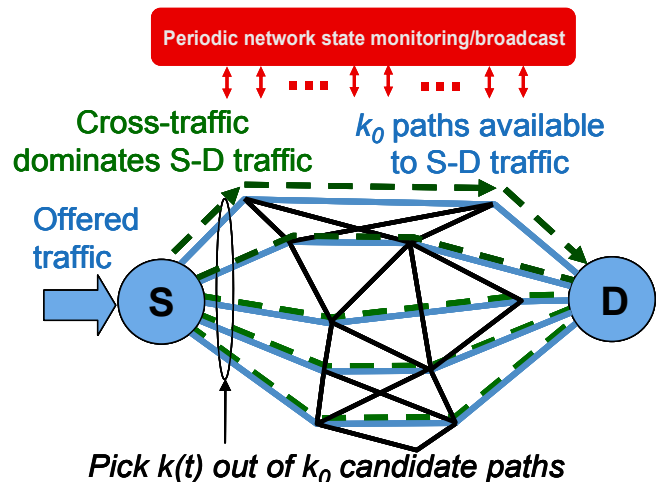


Figure 4. The (slow) centralized network management system periodically broadcasts the path states at regular intervals. The user/scheduler probes $K_p(t) = \min\{K_k, K_0(t)\}$ paths to achieve the desired blocking probability.

At each network update, the scheduling manager identifies the set of paths that are busy, {set of busy paths: K_a }, (therefore also the set of paths that are available, {set of open paths: K_b }, and the total number of paths between s and d , $K_s = K_a + K_b$. Note: we will label the set and the number of paths in the set interchangeably to keep the notation simple. 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. The number of paths in this set is K_p . If all K_p paths are unavailable upon probing, the flow is blocked. For any $t > 1/\mu$, the average service time of a flow, 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 τ must be $< 1/\mu$.

A. Analysis of Blocking Probability of Algorithm 3

The following are the set of parameters used in this section to obtain the network performance:

K_s : total number of paths between a source-destination pair

K_b : number of paths available at the last update

K_a : number of path occupied at the last update

$K_T(t)$: subset of K_b paths taken by through traffic at time t (arrived since the last update)

$K_o(t) = K_b - K_T(t)$: the remaining subset of K_b that may be open or occupied by cross traffic at time t

$K_p(t)$: number of paths that are probed

K_k : fixed probing threshold - If $K_o(t) < K_k$, then $K_p(t) = K_k$. Otherwise, $K_p(t) \leq K_o(t)$, with exact value dependent on the desired blocking probability β .

$P_{C|k}(t)$: Given $K_o(t) = k$, this is the probability that a given path in the set of k paths has been taken by a cross arrival since the last update

τ : update interval

β : blocking probability of a through arrival

ρ_i : offered load due to through traffic, $\rho_i = \lambda_i/\mu$

ρ_c : offered load due to cross traffic, $\rho_c = \lambda_c/\mu$

Specifically, Algorithm 3 operates according to the following procedure. 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. If $K_o(t) > K_k$, only K_k paths are probed. Blocking of an arrival occurs when all $K_p(t)$ paths probed are occupied.

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:

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

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 $e^{-\lambda_c t h}$. Thus, the probability of a path being taken by cross traffic since the last update is:

$$P_{C|h}(t) \approx 1 - e^{-\lambda_c t h}$$

The blocking probability, $\beta(t)$, can be expressed as:

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

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). 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 and $K_s = K_b + i + j$. 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, given the paths taken by through traffic is known. The first system with through traffic is modeled as an $M/M/m/m$ 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 the independence assumption of cross traffic on all links, we can model the cross traffic on each link as an $M/M/1/1$ system. Each link is a single server queueing system that drops an arrival if there is already one arrival in the system. If the offered load is ρ_c , the blocking probability for the link is $p = \rho_c / (1 + \rho_c)$, 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 :

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

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 a slightly pessimistic 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:

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

Substituting the distributions of K_b and $K_T(t)$:

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

PERFORMANCE OF CANDIDATE ALGORITHMS

In this section we comment on the performance of each algorithm and discuss the relative merits of the algorithms for a realistic network.

Algorithm 1.

Figure 5 shows the number of paths required by *Algorithm 1* to achieve a given blocking probability for various network

loads. 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. If network economy is an issue, the network must also serve pre-emptable scheduled flows as a second class service. 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 λ_t in Figure 5 should be upper-bounded by ~ 1.4 (normalized load per path ~ 0.3).

Algorithm 2.

Figure 6 shows the required number of paths K_s between source-destination node pairs for *Algorithm 2* (denoted as K_{CT}) as a function of the required number of paths K_s for *Algorithm 1* (denoted as K_{TT}). 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 lack of network state information at the source node other than the average loading of the network. In *Algorithm 3*, this problem is alleviated by periodic broadcast of network state information.

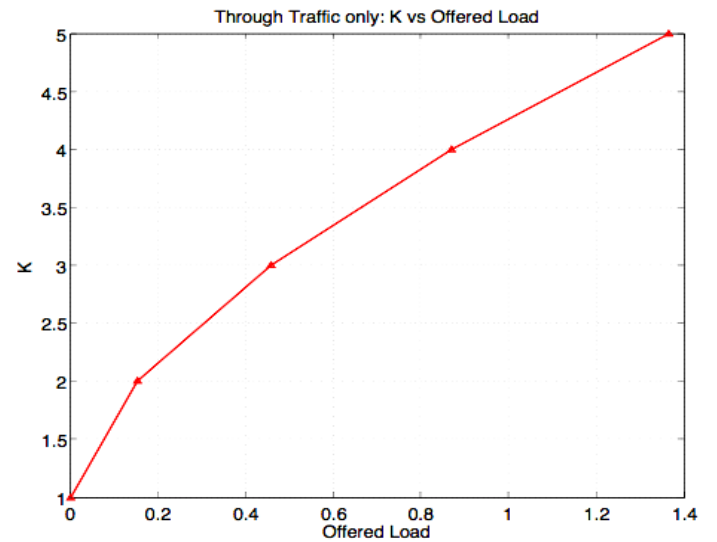


Figure 5. Number of paths K_s needed to maintain $P_B = 10^{-2}$, as a function of offered loading ρ_t for *Algorithm 1*.

^{††}That is the reason in our recent work [6,28], we do not allow cross traffic in the WAN.

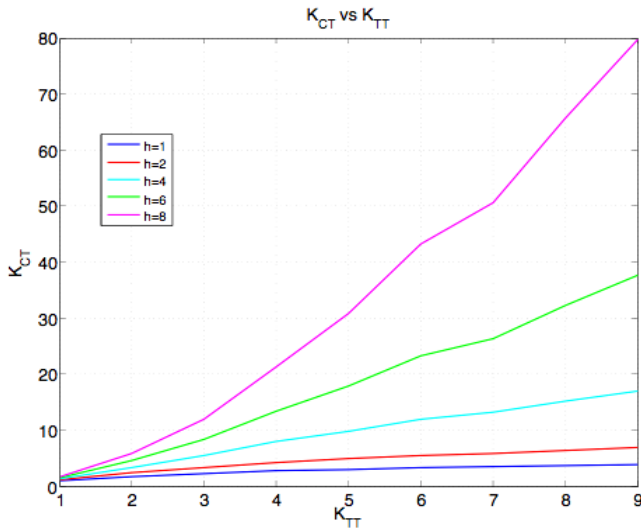


Figure 6. Required number of paths for *Algorithm 2*, K_{CT} as a function of required number of paths K_{TT} for *Algorithm 1* for the same offered load with $P_B = 10^{-2}$.

Algorithm 3.

With *Algorithm 3*, the information right after 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 7 shows that the update interval should be no longer than ~ 0.25 of the expected transaction service time if the update information is of significant value.

Figure 8 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, fewer paths are needed if no cross-flow traffic is allowed into the network. 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.

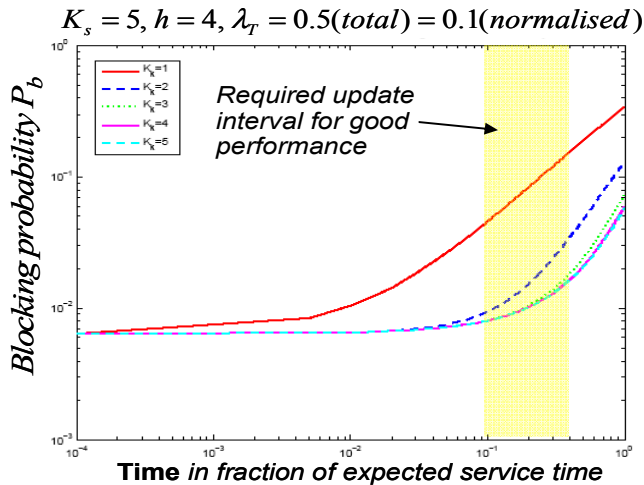


Figure 7. Blocking probability vs. fraction of mean service time for *Algorithm 3*. $\lambda_t = 0.5$, $\lambda_c = 0.1$, $K_s = 5$

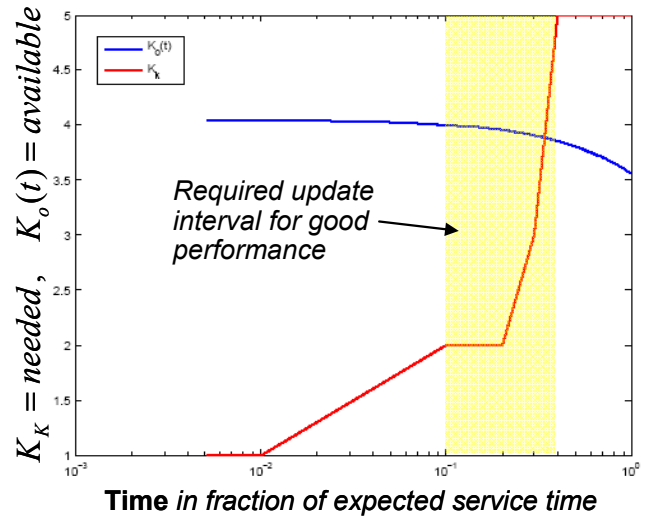


Figure 8. $K_o(t)$ and K_c vs. mean service time for *Algorithm 3*, $\lambda_t = 0.5$, $\lambda_c = 0.1$, $K_s = 5$

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.

V. CONCLUSION

In summary, this paper addresses online OFS network management and control of future high-performance networks serving applications with stringent delay requirements. *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. Therefore, 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..

REFERENCES

- [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. 2146–2168, 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. Médard, "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. Médard, "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.