A Distributed Algorithm for Least Constraining Slot Allocation in MPLS Optical TDM Networks

Hassan Zeineddine and Gregor V. Bochmann,
University of Ottawa,
Ottawa, ON-K1N 6N5

Abstract - **In this paper, we propose a distributed approach for the least constraining slot allocation scheme in all-optical TDM networks (LC) that was introduced in a previous work. The driving force behind our proposal is the employment of the LC scheme in a GMPLS context. After describing the basic data model and messaging parameters, we focus on defining an efficient LC resource status update scheme, which is essential to achieve compatibility with GMPLS' periodic link state update standards. Basically, we reduce the rate of resource status updates from once per call to once per few calls, and measure the impact on network performance.**

I. INTRODUCTION

In a previous work [1], we proposed the least constraining slot (LC) scheduling scheme as a novel slot reservation approach in all-optical TDM mesh networks without buffering [2,3], which are synchronized on slots without synchronization of frame boundaries. Thus, a time sliced traffic segment that travels a route consisting of several links may be carried by slots that have different positions within the frames on the respective links. LC reduces call blocking in these networks to an optimal rate close to what can be achieved with full buffering. The "constraint" in LC is the number of fixed routes that might use a time slot at a given point in time. The algorithm selects the least constraining slots on the route, hence the name "least constraining slot". Comparing its performance to the first fit (FF) approach [2], and FF with optical timeslot interchangers (OTSI) [4], the LC approach provided a performance gain close to the FF approach with OTSI, but at a reduced complexity close to FF without OTSI. The reported gain was consistent under uniform and non-uniform traffic distribution. In addition, we found that the LC approach outperformed the least loaded (LL) approach in multi-fiber environments [5]. Thus, LC proved to have an edge over LL, since the former is not restricted to multi-fibers networks as is the latter. As a general conclusion of our previous study, the LC approach provided close to optimum performance in optical TDM networks with no buffering.

In this paper, we propose a distributed solution for the least constraining slot reservation scheme. Our aim is to employ LC in a Generalized MPLS context [6]. We define the nodal database and basic parameters to be added to GMPLS' reservation and signaling messages. To comply with GMPLS periodic link state updates, we intend to reduce the LC resource status update rate to a level that matches GMPLS standards and still maintains close to optimum performance. In the subsequent sections, we review the LC algorithm and describe the elements of our distributed scheme and how it complies with GMPLS. Before concluding this work, we discuss the effect of reducing the rate of resource status updates on the network performance.

II. LC ALLOCATION SCHEME

Before describing the basic steps of the LC approach, we should clarify the nomenclature used in [1] to provide a better understanding of the presented concepts. Route, route-slot and link-slot are essential concepts used in describing the LC approach.

A network route is a series of unidirectional links interconnected through intermediate nodes from a given source node to a given destination node. Two routes are considered intersecting if they have at least one link in common. A node transmits data into a link in the form of repeating frames of N equal timeslots. Due to link propagation delay, frame alignment is not preserved along the route. Considering link AB, a traffic segment forwarded on a given timeslot at egress node A might be intercepted on a different timeslot at ingress node B. Thus, a timeslot is better identified with reference to a link; we use the term link-slot AB_x to describe timeslot x on link AB. There is no need to mention the corresponding wavelength since only one wavelength plane is considered in this study. Formally speaking, a link-slot is a timeslot on a link with reference to the local clock of its egress node.

In general, a transmitted traffic segment from source node S to destination node D travels through different links along a fixed route, and hence occupies a series of different link-slots. For instance, if A and B are two intermediate nodes between S and D, a series of link-slots would be described as $SA_x\ AB_y\ BD_z$. Knowing the delay of each link, an intermediate link-slot UV_j corresponds to a source link-slot SA_i according to the general rule $j = (i + d_{SU}) \bmod N$, where d_{SU} is the total delay of all links from node S to node U. Thus, knowing the fixed route between a source-destination pair and all associated link delays, we can easily derive the entire series of link-slots for a given link-slot at the source. In this case, we describe the series $SA_x\ AB_y\ BD_z$ with the simple notation $\overline{SD_x}$, which we call a route-slot. The upper bar is essential to differentiate it from a link-slot. A route-slot $\overline{SD_x}$ is considered available if all its constituent link-slots are

available; otherwise, $\overline{SD_x}$ is unavailable. In a single fiber environment, a link-slot is available if it is not reserved. On the other hand, in a multi-fiber case, a link-slot is available if it is free for at least one of the link fibers. To make our approach generic, we develop it based on a multi-fiber environment, and apply it to a single-fiber network as a special case.

The exercise of allocating resources, for a communication request, from node *S* to *D* is to find and reserve an available route-slot $\overline{SD_x}$ along a fixed route, which is assumed to be given.

A. Definitions

If a link-slot XY_j is part of a route-slot $\overline{SD_i}$, we write:

$$XY_j \text{ in } \overline{SD_i}, \text{where } j = (i + d_{SX}) \bmod N . \quad (1)$$

Considering *M* fibers per link, we define the link-slot availability A_{XY_j}, an integer between 0 and *M*, to be the number of fibers on which XY_j is free. If A_{XY_j} is equal to zero, then XY_j is unavailable. Furthermore, we define the availability $A_{\overline{SD_i}}$ of a route-slot to be equal to the minimum availability A_{XY_j} among its constituent link-slots,

$$A_{\overline{SD_i}} = \underset{XY_j \text{ in } \overline{SD_i}}{MIN}\left(A_{XY_j}\right). \quad (2)$$

Knowing the associated fixed route of each source-destination pair, we derive the set Ω of all possible route-slots in the network. We define Ω_{XY_j} to be a subset of Ω consisting of all route-slots that contain link-slot XY_j.

$$\Omega_{XY_j} = \left\{\overline{SD_i} \in \Omega \;\middle|\; XY_j \text{ in } \overline{SD_i}\right\}. \quad (3)$$

We further define Ω'_{XY_j} to be a subset of Ω_{XY_j} consisting of all route-slots whose availabilities are equal to A_{XY_j}.

$$\Omega'_{XY_j} = \left\{\overline{SD_i} \in \Omega_{XY_j} \;\middle|\; A_{\overline{SD_i}} = A_{XY_j}\right\}. \quad (4)$$

The purpose of Ω'_{XY_j} is to identify all route-slots whose availabilities are decremented when XY_i is reserved.

We designate the constraint of link-slot XY_i to be the sum of the availabilities of all route-slots belonging to Ω_{XY_j}.

$$W_{XY_j} = \sum_{\overline{SD_i} \in \Omega_{XY_j}} A_{\overline{SD_i}} . \quad (5)$$

In a single fiber environment, $A_{\overline{SD_i}}$ becomes a binary variable showing whether the route-slot is available (1) or not (0); and hence, W_{XY_j} would reflect the number of available route-slots containing XY_j. In other words, it indicates the number of routes that can potentially use the designated link-slot.

Last, we define the constraint of a route-slot to be equal to the total constraint of all its constituent link-slots:

$$W_{\overline{SD_i}} = \sum_{XY_j \text{ in } \overline{SD_i}} W_{XY_j} . \quad (6)$$

B. Allocation Principle

It is essential to reserve a link-slot which has the lowest interference with other intersecting route-slots, i.e. having the lowest constraint. This keeps more available route-slots in the network, hence improving the blocking rate for subsequent communication requests. Thus, the route-slot that has the lowest constraint $W_{\overline{SD_i}}$ would be the best choice on a given route between *S* and *D*. In this case, only a minimal number of route-slots in the network become unavailable when serving a given call.

C. Constraint Update

After identifying the best route-slot, all constituent link-slots are reserved. Consequently, the constraint of each link-slot in each route-slot in Ω'_{XY_j} is modified according to the algorithm, shown in Fig. 1.

```
foreach XY_j in SD_i do
    W_XYj := W_XYj − 1
    foreach RT_n ∈ Ω'_XYj do
        if RT_n ≠ SD_i
            foreach UV_k in RT_n do
                W_UVk := W_UVk − 1
```

Fig. 1: Constraint update algorithm

By definition (4), Ω'_{XY_j} contains all route-slots whose availabilities are decremented due to a reservation of XY_j. For instance, when reserving XY_j in a single fiber environment, all route-slots in Ω'_{XY_j} become unavailable, and accordingly, their availabilities flip from 1 to 0. Therefore, the constraint of their constituent link-slots must be decremented since a link-slot constraint is the sum of the availability of the intersecting route-slots.

Finally, the same algorithm is repeated when freeing resources, but the constraints are increased instead**.**

III. DISTRIBUTED APPROACH

Aiming to make the LC scheme applicable in a GMPLS context, we should define a distributed algorithm that blends well with GMPLS protocols. In GMPLS, each node has a database and exchanges link state information via update messages based on the Open Shortest Path First (OSPF) [7] or Intermediate System to Intermediate System (IS-IS) routing protocols [8]. For connection reservation, GMPLS uses the Resource Reservation Protocol with Traffic Engineering (RSVP-TE) [9] or the Constraint-Based Routing Label Distribution Protocol (CR-LDP) [10]. Both reservation protocols require two phases: label request phase issued by the source node and label response phase issued by the destination.

The distributed LC approach requires three major components: a database schema at each node, a reservation protocol, and a link state update protocol.

A. Node Database

Each node in a network employing the distributed LC approach maintains a database containing basic information essential for the decentralized reservation process. Basically, for each outgoing link *XY* at a node *X*, two lists must be maintained: Links Info List (LIL) and Link-Slots Info List (LSIL). LIL has entries for each link in the network that shares a route with *XY*. A LIL's entry corresponding to link *UV* has the following structure:

- Total delay *d* in timeslot unit between nodes *U* and *X*. if *U* is upstream from *X*, the delay is a positive integer; otherwise, the delay is a negative.
- Common Route-Slots List (CRSL) containing entries for all route-slots that have link *XY* and *UV* in their paths. Each CRSL's entry stores the route-slot's availability and a set of constituent link-slot ids.

LSIL has an entry for each link-slot on *XY*, which contains the following data:

- Availability
- Constraint

B. Reservation process

Although the aim is to extend the existing RSVP-TE or CR-LDP protocols, we define a new set of messages that can be integrated later with the corresponding messages in these protocols just to avoid lengthy technical discussions that go beyond the scope of this paper. The distributed LC approach uses the following messages during the slot reservation process:

- Request (REQ): it contains source and destination node ids, the cumulative delay, and a route-slot information list (RSIL) which has the same structure as an LSIL. The content of the REQ can be integrated with RSVP Path message.
- Response (REP): it contains source and destination node ids, a selected route-slot, the cumulative delay, and a link-slot availability list (LSAL) indexed by link-slot. The LSAL contains the availabilities for a selection of link-slots. These parameters can be integrated with RSVP Resv message.
- Release (REL): it contains destination node id, and a link-slot. It can be integrated with RSVP Resv Teardown message.
- Negative Acknowledgment (NACK): it is used to inform the source of a denied request. This acknowledgment can be realized by using RSVP Path Error message.

During a reservation process, the following steps are performed:

1. The source node sends a REQ to the destination on a predetermined route. It initially sets the REQ's RSIL to the outgoing link's LSIL.
2. An intermediate node receives the request message, and performs the following steps before forwarding the received message to the next node on the route:
   i. Identify matching link-slots on the outgoing link by using the cumulative delay in the REQ.
   ii. Add the link-slot constraints in the outgoing link's LSIL to the corresponding route-slot constraints in REQ's RSIL.
   iii. Set the availability in each entry of the RSIL to the availability of its corresponding link-slot only if the latter value is less than the former.
   iv. Add the corresponding delay *d* in the LIL to the cumulative delay in the REQ.
3. When a destination node receives the REQ, it sends a REP to the source node after setting the REP's route-slot field to the lowest weighed route-slot in the REQ's RSIL. It also sets the REP's cumulative delay to the REQ's cumulative delay.
4. When an intermediate node receives the REP, it does the following before forwarding the received REP to the next node on the reverse route to the source.
   i. Deduct the corresponding delay *d* in the LIL from the REP's cumulative delay.
   ii. Identify and lock the link-slot that matches the selected route-slot by referring to the cumulative delay.
   iii. Insert the availability of the corresponding link-slot to the REP's LSAL

When the REP message reaches the source node, the node starts transmitting on the reserved route-slot. After completing the communication process, the source sends a REL message towards the destination to free all resources, which were locked for serving the communication request.

A request for communication can be rejected either during the REQ phase or the REP phase. If no matching resource is available during a REQ phase, the REQ message is dropped and a NACK is sent back to the source. In this case, the connection is considered blocked. On the other hand, if two REPs on two intersecting routes require the same available link-slot on the intersection link, the corresponding intermediate node locks this link-slot to the first arriving REP and drops the late one. It sends a NACK to the source of the

dropped REP and a REL to its destination in order to free the locked resources. In this case, the connection is not considered blocked since the source node can retry with another REQ. Further details on the rejection scenarios during the REQ and REP phases can be found in [11].

C. Resource Status Update

With every established or released connection, the constraints and availabilities of corresponding resources change across the network. Therefore, a resource status update scheme is required to keep the databases of all nodes up to date. An update scheme can be instant or periodic. An instant update is broadcasted by the source node upon route-slot's reservation or release. On the other hand, a periodic update is frequently broadcast by all nodes like the OSPF link state update mechanism. In both update schemes, we employ an update message (UPD) similar to the OSPF Update message. However, we append an LSAL as an extra parameter.

While simulating irregularities and errors in the centralized scheme, we explicitly forced our simulated algorithm to skip the constraint update module for n successive calls before executing it at the 1^{rst} call after n. We noticed that network performance remained close to optimum for relatively large n. It basically means that one resource status update every t period of time could maintain close to optimum performance and significantly reduce the associated signaling cost.

Instant Resource Status Update

The following steps occur during an instant resource status update process:

1. The source node notifies all nodes in the network about the reservation of link-slots by broadcasting a UPD message containing the LSAL that was originally carried by the REP.
2. Each node receiving the notification performs the following steps for each reserved link-slot in the LSAL:
   i. Identify the outgoing link that shares a common route with the reserved link-slot if any, by referring to the LIL.
   ii. Identify the corresponding local link-slot by using the total delay from the reserved link's upstream node to this local node.
   iii. Identify the route-slot joining the reserved link-slot with the corresponding local link-slot. This can be achieved by referring to the CRSL.
   iv. If the availability of the route-slot identified in the CRSL is equal to the availability of the reserved link-slot, reduce the route-slot availability and the constraint of the corresponding local link-slot.

Periodic Resource Status Update

In order to implement a periodic resource status update, the following steps are essential:

1. At a fixed time interval t, every node in the network compiles an LSAL and appends it to a UPD message before broadcasting it to the network. A compiled LSAL contains only link-slots availabilities whose values have changed since the previous notification.
2. Each node receiving the notification performs the following steps for each link-slot in the LSAL:
   i. Process the first 3 steps of the instant update case.
   ii. If receiving the first notification for a particular route-slot after the down period t, set the availability of the route-slot identified in the CRSL to the availability of the considered link-slot. Otherwise, execute this step only if the link-slot's availability is the lowest of both values. Depending on the resulting change Δ in the route-slot availability, the constraint of the corresponding local link-slot should change accordingly. If Δ is positive, increase the link-slot constraint by Δ; otherwise, decrease it by $|\Delta|$.

IV. Simulation Results

In this section, we discuss the performance of the distributed LC approach under various status update rates. Our observations are based on simulation results plotted with 95% confidence intervals.

The simulation experiments are based on the 14-nodes 21-link NSFNET network topology [12]. A link between two nodes consists of dual unidirectional fibres with a fixed capacity of 10 timeslot channels per fibre. Fixed shortest path routing is used to derive paths between all source destination pairs. Each path serves up to 10 concurrent connections at the granularity of a transmission channel, i.e. one timeslot per link along the path. Each simulation is repeated for 30 runs. Calls arrive according to a Poisson process, and lasts for an exponentially distributed period.

We study our scheme under two different traffic distributions among source-destination pairs, uniform and non-uniform. In the uniform traffic case, every pair is chosen at random with equal constraint and hence having the same traffic load in Erlang (mean arrival rate × mean holding time). In the non-uniform case, source-destination pairs have different constraints to achieve non-uniform traffic distribution.

Fig. 2 shows the performance of the LC approach for different status update rates. Best performance is obtained for instant updates, that is, an update after each accepted or terminated call. In the case that an update is only done after 10^5 new accepted calls, we obtain what we call "degraded performance"; this performance is approximately one half of the best-performance level that is attained with instant updates. Performance remains at that degraded level even if we increase the update rate to once per 10^2 calls arriving to the network. However, if the update rate is once per 10 calls, we obtain best performance as in the case of instant updates. As a generalization, we consider that the performance is unaffected if the update rate is greater than or equal to λ/α where λ is the call inter-arrival rate, and α is a coefficient

related to the network size which is close to 10 for the NFSNET.

Fig. 3 is a chart that shows samples of blocking probability collected over several short periods of 10 calls each. To reduce statistical variations caused by sampling over short periods, the same simulation is repeated 10000 times with a high load of 120 Erlang. The employed status update rate is once per 500 calls after an initial period (not shown in the diagram) of instant updates. We notice an initial transition period of gradual performance degradation reflected in the early samples. The transition is from the best-performance level to the degraded performance level. The first couple of samplings are close to the best performance rate of 0.027. If we average out the statistical variations after the transition period, the worst performance rate seems to stabilize at a fixed level of 0.035 on average. The chart also shows that subsequent (single) status updates do not reproduce the best performance rate observed earlier. To discuss these results further, we define the following:

- ω_t: is the list of all recorded route-slots constraints in the network. The constraint values are based on link-slot constraints that are recorded in nodal databases.
- $\acute{\omega}_t$: is the list of all actual route-slots constraints in the network. The constraint values are based on actual link-slot constraints that are not recorded in nodal databases.
- *Low*(*SD*, ω_t): is a function that returns the route-slot on route *SD* that has the lowest constraint according to ω_t.

In the case of instant updates, ω_t should always be equal to $\acute{\omega}_t$ at any time *t*; i.e. $\omega_t = \acute{\omega}_t$, and hence *Low*(*SD*, ω_t) = *Low*(*SD*, $\acute{\omega}_t$) for all routes at any point in time. This equation is essential for a perfect route-slot allocation pattern and best network performance. Starting from a perfect LC allocation pattern and stopping all further updates, ω_t and $\acute{\omega}_t$ would break ties after the first allocated or de-allocated call; and ω_t is said to be outdated. However, the equation *Low*(*SD*, ω_t) = *Low*(*SD*, $\acute{\omega}_t$) might still hold for a majority of routes during the first few allocated or de-allocated calls. As long as this equation holds for the routes at which all subsequent calls arrive, the system would allocate the same route-slots that would be chosen in the case of instant updates. Thus, the perfect LC route-slots allocation pattern in the network is maintained, and hence best performance is preserved. As soon as a call arrives at a route *SD* where *Low*(*SD*, ω_t) ≠ *Low*(*SD*, $\acute{\omega}_t$), the resulting allocation pattern becomes imperfect; and hence performance starts to degrade. The length of the best performance period preceding the degraded performance is given by α/λ. Note that α depends on the probability *P/n(n-1)*; where *n* is the number of nodes in the network, and *P* is the probability of having *Low*(*SD*, ω_t) ≠ *Low*(*SD*, $\acute{\omega}_t$) for a given route *SD* (note that *n(n-1)* is the number of routes in the network). *P* is relatively small during the first few calls and increases gradually with every allocated or de-allocated call as each route-slot affects the constraints of its intersecting route-slots. During the best performance period, the constraints in $\acute{\omega}_t$ will always be based on a perfect LC allocation pattern. As a result, if (single) updates occur at a period shorter than or equal to the best performance period, ω_t will always be based on a perfect LC pattern; best performance is continually maintained. On the other hand, if (single) updates occur at a longer period, ω_t will most likely be based on an imperfect allocation pattern leading to degradation of performance.

Regardless of the update rate, network performance is at the degraded level as long as the update interval is longer than the best-performance period. Note that if the route-slot allocation pattern becomes imperfect it reflects an imperfect $\acute{\omega}_t$. After an update, $\acute{\omega}_t$ gets copied to ω_t which would emphasize the pattern's imperfection. Thus, further updates emphasize rather than fix imperfection; and hence, the irrelevance of update rates to the performance degradation level is now clear.

Although the route-slot allocation pattern is imperfect, the performance level is still better than the performance level of the FF approach. Note that an outdated ω_t still imposes an order that the system follows when allocating route-slots. This order is the result of the most recent LC update. It actually gives different priorities to the route-slots in all routes according to the constraints collected by the last update. Note that the resulting priority order for different routes is not arbitrary but rather synchronized based on the LC update. This synchronization between different routes is the essence behind the degraded performance level which is better than the worst case scenario of the FF approach. As an analogy, a synchronized traffic light system based on an outdated traffic pattern would still manage traffic better than a chaotic arbitrary system.

In a multi-fibers environment, an imperfect route-slot allocations pattern can still produce a performance similar to what is obtained with a perfect pattern (see Fig. 4). The improved performance of an imperfect pattern in a multi-fibers environment is mainly due to the additional number of fibers. Although *P* in a multi-fibers environment is smaller than in the single fiber case, it does not justify the difference in performance. Its effect will be limited to slightly increasing the perfection period. Regardless of this period, the pattern would eventually become imperfect after few calls without updates. However, the priority order that remains in effect still produces close to optimal performance as shown in Fig. 4. Therefore, we conclude that network performance metrics resulting from imperfect and perfect allocation patterns converge as an effect of extra fibers.

V. Conclusion

In a previous work, we proposed the least constraining slot reservation approach (LC) for all-optical TDM networks. In this paper, we designed a distributed LC scheme in an attempt

to make it applicable in GMPLS networks. After specifying the node database, we defined new parameters that need to be added to the RSVP-TE or CR-LDP messages. In addition, we developed two different resource status update schemes: instant and periodic. The major challenge was to incorporate the LC resource status update into GMPLS, which relies on OSPF or IS-IS link state update mechanisms. Since GMPLS' relies on global periodic updates, we have to skip a number of calls before invoking the LC resource updates. We showed by simulation that an update rate greater than or equal to λ/α maintains close to optimal performance; where λ is the call inter-arrival rate, and α is a coefficient related to the network size which is close to 10 in NFSNET. For lower update rates, performance degrades to a fixed level but does not converge to the worst performance level reported with the first fit (FF) approach; hence, stopping all subsequent updates throughout the network lifetime after a brief period of instant updates produces a performance level as good as any update rate less than λ/α. In multi-fiber environments, the update reduction has no significant effect on performance regardless of the rate. In this case, stopping the instant updates at an early stage of the network operation does not affect performance, and hence the associated signaling bandwidth is spared. As a general conclusion, the distributed LC scheme produces a close to optimal performance in a GMPLS optical TDM network after slightly extending the reservation protocol and not changing the rate of link state updates.

REFERENCES

[1] H. Zeineddine and G. V. Bochmann, "Least Constrained Slot Allocation in Optical TDM Networks," Proc. IEEE Wireless and Optical Communications Networks, July 2007, pp. 1-5.

[2] J. M. Yates, J. P. R. Lacey, and D. Everitt, "Blocking in multiwavelength TDM networks," Telecommunication Systems Journal, vol. 12, no. 1, pp. 1-19, August 1999.

[3] H. Zang, J. P. Jue, and J. Mukherjee, "Photonic Slot Routing in All-Optical WDM Mesh Networks," Proc. IEEE Globecom '99, Rio de Janeiro, Brazil, December 1999.

[4] H. Zeineddine, P. He, and G. V. Bochmann, "Optimization Analysis of Optical Time Slot Interchanges in All-Optical Network," Proc. IASTED Wireless and Optical Communications, July 2006, pp. 207-212.

[5] B. Wen, R. Shenai, and K. Sivalingam, "Routing, Wavelength and Time-Slot Assignment Algorithms for Wavelength-Routed Optical WDM/TDM Networks," Journal of Lightwave Technology, vol. 23, No. 9, September 2005.

[6] A. Banerjee, et al., "Generalized multiprotocol label switching: An Overview of Routing and Management Enhancements," IEEE Commun. Mag. (2001) 144–150.

[7] J. Moy, "OSPF Version 2," STD 54, RFC 2328, April 1998.

[8] D. Oran, "OSI IS-IS Intra-domain Routing Protocol," RFC 1142, February 1990.

[9] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels," RFC 3209, December 2001.

[10] B. Jamoussi et al., "Constraint-Based LSP Setup using LDP," RFC 3212, January 2002.

[11] X. Yuan, R. Gupta, and R. Melhem, "Distributed Control in Optical WDM Networks," IEEE Conf. on Military Communications, McLean, VA , October 1996.

[12] B. Mukherjee, "Optical WDM Networks," Birkhauser, 2006.

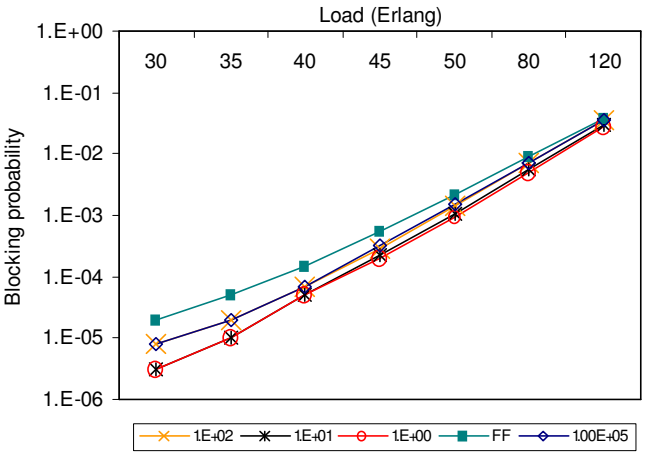

Fig. 2: LC performance for different update rates (once per 1E+x calls)

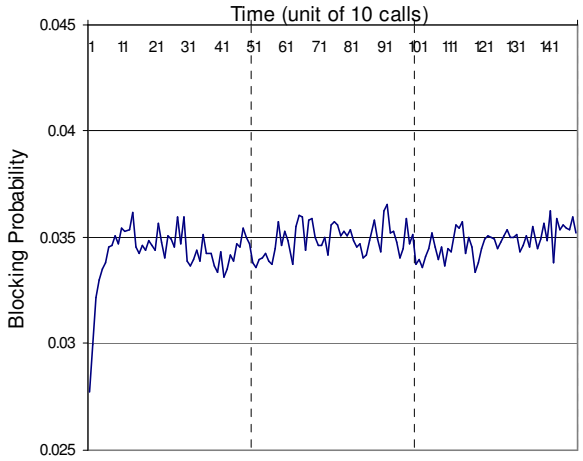

Fig 3: LC performance measured every 10 calls – load is 120 Erlang – (update rate is once every 500 calls)

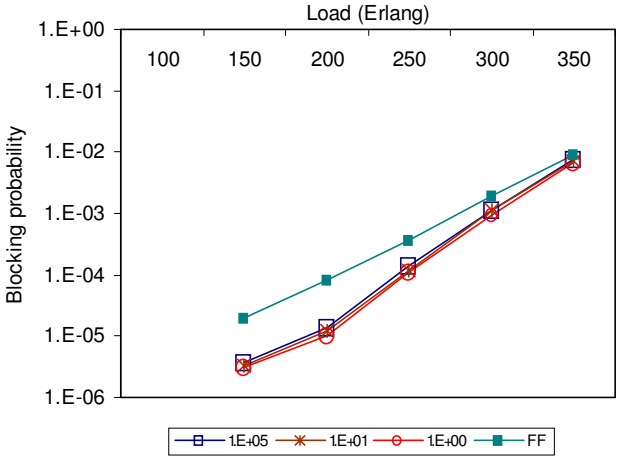

Fig. 4: LC performance for different update rates (once per 1E+x calls) in a 3-fibers network