# Helios: A Broadcast Optical Architecture*

Ilia Baldine[1], Laura E. Jackson[1], and George N. Rouskas[2]

[1] MCNC ANR, Research Triangle Park, NC, USA, ibaldin,lojack@anr.mcnc.org
[2] Department of Computer Science, North Carolina State University, NC, USA, rouskas@eos.ncsu.edu

**Abstract.** In this article we present a new all-optical broadcast LAN architecture and an accompanying signaling protocol. The distinguishing characteristics of this architecture are its fault-tolerant design and its collision-free nature, which allows it to achieve high throughput in a broadcast environment. The flexibility of the design allows different schedulers to be used, which can introduce new features into the network (e.g. multicast and QoS) as well as optimize its behavior for the specific setting in which it is used.

## 1 Introduction

Wavelength division multiplexing (WDM) optical networks are a viable technology for a next-generation network infrastructure that supports a diverse set of existing, emerging, and future applications [8]. WDM bridges the gap between lower electronic switching speeds and ultra high optical transmission speeds. Dividing the enormous information carrying capacity of single mode fiber into a number of channels, each on a different wavelength and operating at peak electronic speed, WDM makes it possible to deliver aggregate throughput on the order of Terabits per second. WDM technology initially was deployed in point-to-point links and has also been extensively studied, theoretically and experimentally, in wide area or metropolitan area distances [7]. Several WDM local area testbeds have also been implemented [5] or are currently under development [6, 1].

In this article we present `Helios` – a WDM all-optical architecture for a local area network and an accompanying signaling protocol. The packet-oriented `Helios` architecture enjoys independence of the number of nodes and the number of supported wavelengths, and relies on scheduled access to the medium, guaranteeing higher utilization. `Helios` is part of a DARPA-funded project aimed at demonstrating the feasibility and potential of optical access networks. This effort is a logical continuation of earlier work performed at NCSU ([11], [12]). Following an overview of the architecture in Sect. 2, we describe the signaling protocol in Sect. 3 and the basic `Helios` scheduling algorithm in Sect. 4.1. We conclude in Sect. 5.

* Supported by DARPA under Contract No. F-30602-00-C-0034 and NSF under Grant No. 9701113.

E. Gregori et al. (Eds.): NETWORKING 2002, LNCS 2345, pp. 887–898, 2002.

## 2 The Helios Architecture

The Helios network employs a passive star coupler (PSC) as a broadcast medium to connect all nodes in the network, making Helios a *single-hop* WDM network. The entire path between source and destination in such a network is entirely optical; no electro-optic conversion of the signal is necessary [9]. Helios uses a smaller number of wavelengths than the potentially large number of nodes. The Layer 3 protocol could be either IPv4 or IPv6.

Communication in a Helios network is made collision-free by a non-preemptive gated scheduling protocol. A single *master* node in the network calculates and disseminates the schedule, while other nodes use this schedule to time the transmission of data to their peers. There are two types of nodes: *candidate* nodes, which are eligible to serve as the master node should the current master node fail, and *slave* nodes, which are not. Such a distinction is necessary because a network will likely be composed of servers and workstations, where the workstations lack the necessary computing resources to perform the master node's duties. Furthermore, workstations may allow low priority user access, making them vulnerable to security attacks that could disrupt the network.

The Helios network utilizes a Fast Tunable Transmitter – Slowly Tunable Receiver (FTT–STR) approach, where *fast* implies low to sub-microsecond tuning times and *slow* implies hundreds of microseconds to tens of milliseconds. For packet transmission and scheduling purposes, the lasers are considered tunable and the receivers fixed. However, in order to balance the load in the network, the receivers may be retuned from time to time, on the order of seconds.

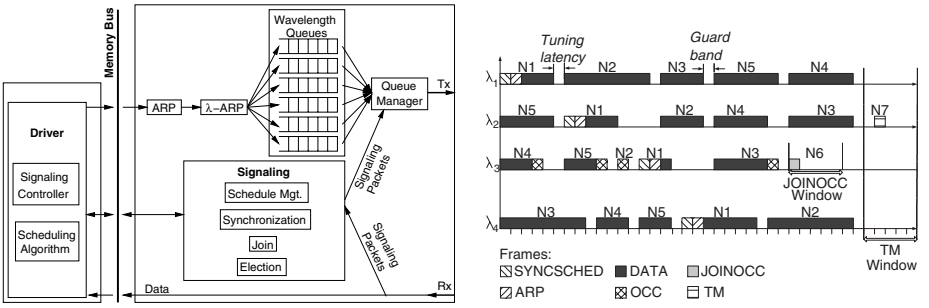
Helios differs from other WDM networks currently under development in several respects: it operates within a broadcast-and-select environment, it is collision-free, and it is packet-switched instead of circuit-switched. At the same time, the Helios architecture provides for such important LAN features as native QoS support and multicast, described in [4] and [13].

### 2.1 High Level Node Design

Figure 1(a) highlights the various hardware, software, and firmware components of a Helios network adapter. The software **Driver** module consists of two sub-modules. The **Signaling Controller** coordinates the operation of all other software and hardware modules. The **Scheduling Algorithm** calculates new schedules based on queue occupancies provided by all nodes in the network; it is called infrequently, in response to changes in the traffic pattern or simply periodically.

In hardware, the **Signaling** module of the adapter contains four sub-modules: **Schedule Management** forms and processes frames related to scheduling, **Synchronization** enables all communication to occur in hard real time, **Join** allows a new node to join a Helios network, and **Election** manages the selection of a new master node when the current master node fails.

The **ARP** and **$\lambda$-ARP** tables enable a Helios node to perform IP-to-MAC address resolution and MAC-to-receive-wavelength resolution, respectively. The




(a) High level node architecture (b) Example superframe

**Fig. 1.** Overview of `Helios` node architecture and superframe organization

master node keeps track of ARP and $\lambda$-ARP mappings and distributes them via ARP frames to all other nodes. Outgoing IP packets are buffered in the **Wavelength Queues** on a per-wavelength basis prior to transmission. The **Queue Manager** serves the wavelength queues in FIFO order and controls which frames are transmitted.

### 2.2 Frames and Superframes

The time required to complete the transmissions of one full schedule in `Helios` is a *superframe*. A superframe further consists of frames, continuous sequences of octets transmitted by nodes on individual wavelengths; Table 1 shows the different frame types. `Helios` uses non-preemptive schedules, thus within each superframe a node transmits on a particular wavelength at most once.

**Table 1.** `Helios` frame types and their function

| | |
|---|---|
| DATA | Carries regular data |
| MDATA | Carries multicast data |
| TM | Measures roundtrip delay to PSC |
| OCC | Transmits queue occupancies to master node (Routine mode) |
| JOINOCC | Transmits queue occupancies to master node (Join mode) |
| SYNCSCHED | Carries scheduling information |
| ARP | Carries MAC to wavelength index mapping ($\lambda$ARP) |
| OAM | Carries error and management information about network state |
| AVAIL | Announces availability of a candidate node to become the master node during scheduler election |

The master node calculates the schedule based on other nodes' packet queue occupancies, which it learns through the OCC frames sent by other nodes during routine network operation. Once calculated, the schedule is then broadcast on each wavelength inside the SYNCSCHED frame, which the master node transmits on every wavelength every superframe. A schedule contains **windows**, or intervals of time, during which a particular node may transmit a frame.

Figure 1(b) shows the position of various frames and windows within a superframe. In this example, N1 is the master node and its receive wavelength is $\lambda 3$. There is a JOINOCC window on $\lambda 3$ (with a JOINOCC frame in it), and there is an attached TM window at the end of the superframe. Two nodes are in different stages of joining the network: N6 is sending a JOINOCC frame containing its queue occupancy information to the master node so that it can be included in the next schedule. Meanwhile, N7 is performing Time Measurement; its TM frame can be seen inside the TM window. Time measurement is the first operation a new node must perform when joining the network, in order to synchronize frame reception and transmission.

## 3 Network Operation: The `Helios` Signaling Protocol

The operation of a node in the `Helios` network is separated into the six different modes shown in Table 2. Following an overview of each mode, we discuss one, Routine Mode, in detail. When the network comes up after having been completely powered down, no master node has yet been designated, no frames are traveling, and no synchronization information is available. The first task is the election of a master node; candidate nodes enter **Election Mode** while slave nodes sleep. The operation of Election Mode assumes that candidate nodes are equipped with slowly tunable receivers; otherwise, a network administrator must designate the master node.

Once a master node has been elected, it circulates scheduling and synchronization information in SYNCSCHED frames. Now other nodes may join the network, by proceeding through the **Time Measurement** and **Join** modes. In Time Measurement, a node calculates its `psc_offset`, the propagation delay to the PSC. All times are measured locally, and the transmissions are done in relation to the PSC time. Since collisions can occur only at the PSC, each node uses its `psc_offset` to ensure that its transmissions reach the PSC at the exact

**Table 2.** Modes of operation in the `Helios` network

| | |
|---|---|
| Time Measurement | a new node measures its propagation delay to the PSC |
| Join | a new node contacts the master node with its bandwidth requirements |
| Election | a candidate node participates in the election of a new master node |
| Routine | a node transmits and receives data and related signaling frames |
| Scheduling | same functions as routine, plus must create and distribute new schedules |
| Error | error detection, report and recovery |

time prescribed by the schedule. After Time Measurement, a node enters **Join Mode**. It informs the master node of its traffic demands via the JOINOCC frame; the master node then calculates a new schedule to include this new demand. The joining node waits to hear the new schedule before beginning normal transmissions.

It is possible for a collision to occur when two or more nodes attempt to join a `Helios` network at the same time. Two nodes assigned to the same listening wavelength could experience a collision during Time Measurement, or two nodes may transmit a JOINOCC frame to the master node during the same JOINOCC window. The protocol includes backoff algorithms to resolve such contention.

After **Join**, a node enters **Routine Mode**, where it remains unless an error condition occurs. The receive hardware extracts the schedule from arriving SYNCSCHED frames and forwards incoming data frames to the driver; meanwhile, the transmit hardware transmits control frames and data frames from its wavelength queues onto the appropriate outgoing wavelengths, according to the current schedule. Among these transmissions is an OCC frame sent to the master node, once per superframe, to communicate the node's packet queue occupancies; the master node uses queue occupancies from all nodes to recalculate the schedule. Unlike Time Measurement and Join modes, Routine Mode is collision-free. The `psc_offset`, first measured during Time Measurement, is also measured periodically during Routine Mode, in a collision-free manner.

### 3.1 Routine Mode: The Receiver State Machine `>routine<`

Figure 2 shows the state machine `>routine<` which governs the receiver's actions during routine mode. A software signal to `>routine<` initiates the transition out of IDLE and into the ROUTINE LISTEN state. When a SYNCSCHED frame arrives, `>routine<` first checks whether its own MAC address (`my_node_ID`) is included in the schedule. If the node has been left out of the schedule, `>routine<` sends the "NOT_IN_SCHED" signal to the Signaling Controller and returns to IDLE. The Signaling Controller then exits Routine Mode and moves to Error mode.

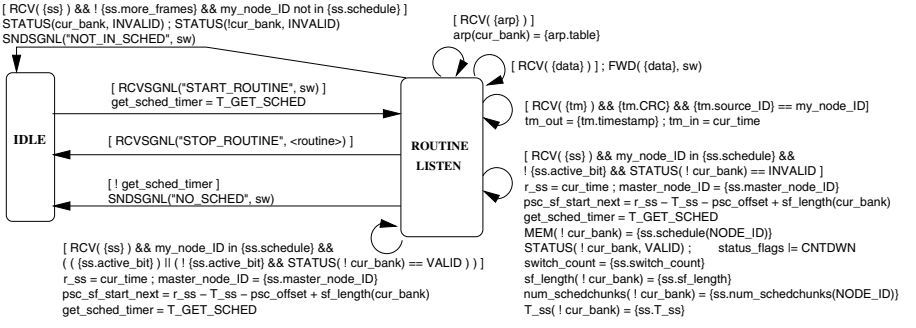



**Fig. 2.** Receiver hardware state machine for routine mode: `>routine<`

If, on the other hand, the node's MAC address (`my_node_ID`) *is* in the schedule, then `>routine<` next checks whether the "active bit" field within the SYNCSCHED frame, called {`ss.active_bit`}, is set. As long as the active bit is set, the node will continue to operate according to the current schedule located in the current memory bank, `cur_bank`. However, if the active bit is not set, then the schedule being disseminated in the SYNCSCHED frame is a newly calculated schedule that will go into effect after `switch_count` more superframes. That is, `switch_count` represents the number of remaining superframes following the current one in which the old schedule will still be used. The value of `switch_count` is obtained from the SYNCSCHED frame.

When `>routine<` encounters a SYNCSCHED frame without the active bit set, it checks the status of the reserve memory bank, `!cur_bank`. If the status is `INVALID`, then all the new synchronization and scheduling information for the new schedule has yet to be copied into the reserve memory bank, `!cur_bank`. After copying this information, `>routine<` sets this bank's status to `VALID`. In this way, `>routine<` doesn't waste effort recopying the new schedule's information into the reserve memory bank several times. That is, if `>routine<` encounters a SYNCSCHED frame without the active bit set but finds the status of the reserve memory bank to be already `VALID`, then it recognizes that it has already copied the new information into the reserve memory bank.

Routine mode ends whenever one of several error conditions occurs. For example, if a SYNCSCHED frame isn't received within the allowed time interval, then it is possible the master node has failed; thus `>routine<` generates the "NO_SCHED" signal and returns to the IDLE state.

## 4 Scheduling

### 4.1 The `Helios` Greedy Scheduling Algorithm

The master node receives an OCC frame containing packet queue occupancies from each node once per superframe. The master node may also receive a JOIN-OCC frame containing packet queue occupancies from a new node joining the network. From this information, the master node builds the $N \times C$ traffic matrix, where $N$ is the number of nodes in the network, $C$ is the number of wavelengths, and entry $a_{ij}$ is the number of slots requested by node $i$ for transmission on $\lambda_j$.

**Table 3.** Example traffic matrix

| | $\lambda_1$ | $\lambda_2$ | $\lambda_3$ | sum |
|---|---|---|---|---|
| $n_1$ | 4 | 1 | 3 | 8 |
| $n_2$ | 2 | 3 | 2 | 7 |
| $n_3$ | 3 | 2 | 1 | 6 |
| $n_4$ | 2 | 3 | 1 | 6 |
| $n_5$ | 1 | 1 | 2 | 4 |
| sum | 12 | 10 | 9 | |

Table 3 shows a traffic matrix for a network of $C = 3$ wavelengths and $N = 5$ nodes.

`Helios` uses a one-pass greedy scheduling algorithm, the pseudocode for which is given in Alg. 1. The algorithm creates a schedule from $t = 0$ forward in time without backtracking, always attempting to schedule the highest priority node on the highest priority wavelength. Higher priority is assigned to nodes (respectively, wavelengths) that have higher corresponding row-sums (respectively, column-sums) in the traffic matrix. In the sample traffic matrix in Table 3, the nodes have been renumbered in order of largest row-sum to smallest, such that $n_1$ has the largest row-sum and $n_N$ has the smallest, with ties being broken arbitrarily. The same was done for the wavelengths: $\lambda_1$ has the largest column-sum and $\lambda_C$ has the smallest. The traffic matrix gives rise to two lower bounds on the schedule length. The maximum column-sum is the *channel bound*; a schedule can be no shorter than the total demand for any one wavelength. The maximum row-sum plus $C$ tuning latencies is called the *node bound*; to meet the demand of $n_1$, a schedule must be at least long enough for $n_1$ to transmit all its traffic and tune to each of the $C = 3$ wavelengths. The maximum of the channel and node bounds is the greatest lower bound on the schedule length.

The original scheduler developed in a previous work at NCSU ([11], [12]) produces schedules very close to the lower bound in length, but requires a prohibitively long runtime. In particular, the original scheduler has a worst-case runtime of $O(CN^4)$. The scheduler developed for `Helios` is a straightforward greedy scheduler that has a worst-case runtime of $O(C^2N^2)$. This speedup is substantial because the number of nodes is expected to be much larger than the number of channels. Moreover, the greedy scheduler can be readily implemented in hardware, resulting in an additional gain in speed. To achieve these gains in speed and simplicity, the new scheduler produces schedules that are not as close to optimal as those produced by the original scheduler. However, the greedy scheduler's results are "reasonably close" to optimal: in simulations with various patterns of network traffic demand, the greedy scheduler produces schedules within 5% of the lower bound, approximately 95% of the time.

The histogram shown in Fig. 3 corresponds to a network of 50 nodes, in which each node determines its demand for each wavelength by drawing from the same distribution (here, equally likely over the set {0,1,...,20}). For each set of traffic demands, we examined the ratio of the length of the schedule generated by the greedy scheduler to the lower bound. The histogram was created from 100,000 replications. The height of each box shows the number of replications in which the ratio fell within the range indicated. For example, nearly 58,000 or 58% of the replications resulted in ratios between 1.00 and 1.01. Furthermore, in 95% of the replications, the new scheduler produced a schedule that was no more than 3% longer than the lower bound (corresponding to ratios between 1.00 and 1.03).

**Algorithm 1** The helios greedy scheduler

```
{* initialize each entry in the schedule to 0 *}
for t = 0 to 2glb do {* schedule length won't exceed 2×greatest-lower-bound *}
  for λ = 1 to C do
    schedule[t][λ] ← 0
  end for
end for
{* initialize remainingDemand to the sum of all the a_nλ's *}
remainingDemand ← 0
for λ = 1 to C do
  for n = 1 to N do
    remainingDemand ← remainingDemand + a[n][λ]
  end for
end for
{* begin scheduling at first slot *}
t ← 0
while remainingDemand > 0 and t < 2glb do {* there is still unmet demand *}
  for λ = 1 to C do
    if schedule[t][λ] = 0 then {* if no task has been assigned to this λ, this slot *}
      n ← 1
      while n ≤ N and (unavailable[n][t] = 1 or a[n][λ] = 0) do
        n ← n + 1 {* find an available node with unfulfilled demand on this λ *}
      end while
      if n ≤ N then
        for i = t to t + a[n][λ] −1 do
          schedule[i][λ] ← n
        end for
        for i = t to t+a[n][λ]−1 + tuneLatency do
          unavailable[n][i] ← 1
        end for
        remainingDemand ← remainingDemand −a[n][λ]
        a[n][λ] ← 0
      end if
    end if
  end for
  t ← t + 1 {* move to next slot *}
end while
```

### 4.2 Multicast

The `Helios` network nodes are equipped with fast tunable transmitters and slowly tunable receivers to form what is known as a FTT-STR architecture. For functions such as packet transmission and scheduling which operate at fine time scales (i.e., on the order of packet transmission times), the lasers are considered tunable and the receivers are considered fixed-tuned. The tunability of optical receivers is invoked only at longer time scales (i.e., on the order of seconds or hundreds of milliseconds) to address the issues of load balancing and multicast. In other words, we distinguish two regions of network operation: during the

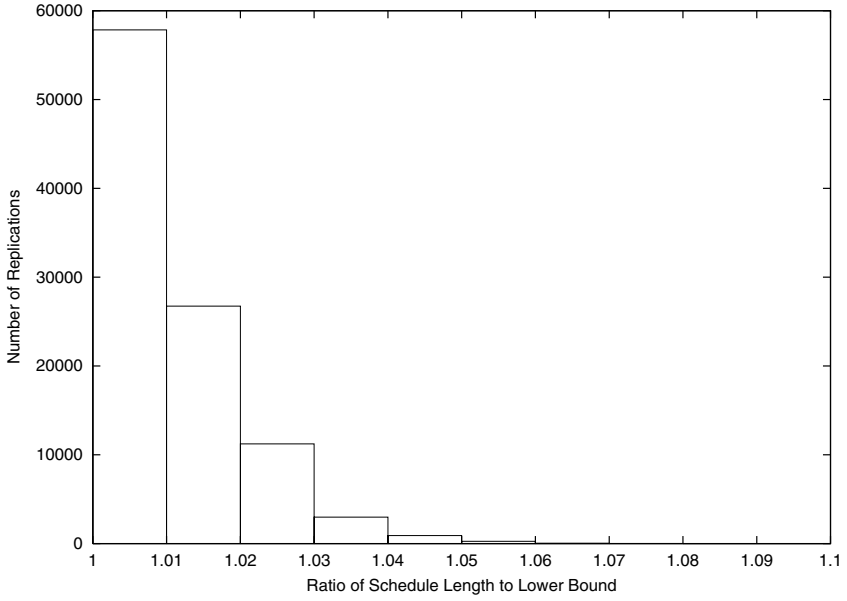



**Fig. 3.** Performance of the greedy scheduler in a `Helios` network of 50 nodes

*normal operation* phase, the optical receivers remain fixed-tuned to their home channels, while during the *reconfiguration* phase [3], the receivers are slowly retuned to new home channels in order to optimize the network for the next normal operation phase.

Let us assume that we have some information regarding the long-term multicast traffic demands in the network, including the number and composition of multicast groups, and let us further assume that this information is collected using the Helios protocol implemented at each node. Then, the problem of supporting multicast traffic in a FTT-STR broadcast WDM architecture is an optimization problem, whereby optical receivers must be assigned home channels such that a performance metric is optimized. The performance metric of interest in `Helios` is the *multicast throughput*, defined as the number of multicast completions per unit time, where a multicast completion refers to the transmission of a multicast packet to all members of its multicast group. We refer to this problem as the *multicast wavelength assignment* (MWA) problem, and we have shown in [13] that it is NP-hard.

The complexity of the MWA problem derives from two conflicting objectives that must be simultaneously satisfied. On the one hand, it is important to balance the traffic load across the different channels, while on the other hand it is desirable to assign receivers in the same multicast group to the same home channel to keep the multicast throughput high (otherwise, a multicast packet has to be transmitted multiple times, once to the home channel of the various receivers in its group). The problem is further complicated by the fact that mul-

tiple groups may not be disjoint, i.e., a given receiver may be part of multiple groups.

We have developed a number of heuristics for the MWA problem, which are described in detail in [13]. Here we provide a summary of their operation. The `Join` class of heuristics starts with each of the $N$ receivers assigned to a separate channel, and repeatedly joins the receivers from two different channels by assigning them to a single channel, until the number of home channels is equal to the number $C, C < N$ in the network. The `GreedyJoin` heuristic applies a greedy rule in joining two sets of receivers, while the `RandomJoin` heuristic randomly joins two sets at each step. The `Split` class of heuristics starts with all $N$ receivers in the network assigned to a single home channel, and then repeatedly selects one receiver to assign to one of the other $C - 1$ channels. The `Join` class and `Split` class of heuristics take advantage of the *monotonicity* properties of the multicast throughput that were first derived in [10]. The `MLPT` heuristic takes a different approach. It first uses the LPT (Largest Processing Time) scheduling algorithm, which provides good load balancing, to come up with an initial wavelength assignment, which it then improves through an iterative approach. Based on a wide range of results in [13], the `GreedyJoin` heuristic appears to provide the best approach for the MWA problem.

### 4.3 DiffServ Support in the `Helios` Architecture

The basic `Helios` scheduling algorithm is appropriate for best-effort traffic but does not provide any QoS guarantees. We have modified this scheduling algorithm [4] to provide native support for the differentiated services (DiffServ) architecture currently being standardized by the IETF. Providing bandwidth and/or delay guarantees in a multiwavelength environment is an inherently complicated task, due to the need to coordinate packet transmissions among the nodes across multiple wavelengths while at the same time attempting to meet packet deadlines; the problem becomes all the more difficult when the transmitting nodes have to account for non-negligible tuning delays. We provide a brief summary of the scheduling algorithm here; details and numerical results are available in [4].

The algorithm consists of two steps. First, an initial schedule is built based on traffic reservations for the two classes of DiffServ traffic that require bandwidth and/or delay guarantees, the Expedited Forwarding (EF) class and the Assured Forwarding (AF) class. This schedule is such that all nodes can meet the QoS guarantees for their EF and AF traffic. This initial schedule is then extended to assign transmission slots for best-effort (BE) traffic, using an algorithm that ensures two important properties in the final schedule: first, that the QoS of the EF and AF traffic is not compromised for any node; and second, that best-effort transmissions are assigned to the various nodes in a *max-min* fair fashion. This latter property guarantees that the excess bandwidth in a `Helios` network is allocated fairly among the network flows. Another important feature of our guaranteed-service scheduling algorithms is that they require only small changes to the basic `Helios` scheduling algorithm. Numerical results in [4] using our

WDM simulator (see below) indicate that the algorithm works as expected and can provide QoS guarantees compatible with the DiffServ framework.

A significant contribution of our work was the implementation of a highly extensible simulator for evaluating the performance of the scheduling algorithms. Our simulator builds upon the functionality provided by the DiffServ model contributed by Nortel Networks to the popular simulator tool `ns-2`. Before our work, `ns-2` lacked support for WDM (i.e., multi-channel) links. Our WDM simulator was integrated into `ns-2` by mapping a model of a `Helios` node into an `ns-2` topology. The details of the mapping can be found in [4], while the computer code is available at [2] and can be easily incorporated into an existing `ns-2` installation. We believe that our simulator addresses an important need and we hope that it will be useful to other researchers in the field.

## 5 Conclusion

In this article we have presented a WDM all-optical broadcast architecture for a local area network with an accompanying signaling protocol and control algorithms. We've demonstrated how elements of DiffServ (QoS) and multicast can be easily incorporated into the architecture, both essential features for local area networks of the future.

We believe the `Helios` architecture to be a viable concept for all-optical networks of the future. Features such as fault-tolerance, the ability to support more nodes than wavelengths, and scheduled gated access to the medium combine to make this architecture a flexible framework into which, by replacing only the scheduler, new features can easily be incorporated. Our work on `Helios` continues. We plan to implement an emulation of the protocol running on commodity hardware to test various approaches to scheduling and signaling, in order to validate the concept even further.

## References

1. The NGI Helios project. In *http://helios.anr.mcnc.org/*.
2. WDM support in ns-2. In *http://www.csc.ncsu.edu/faculty/GRouskas/NS/*.
3. Ilia Baldine and George N. Rouskas. Traffic adaptive WDM networks: A study of reconfiguration issues. *IEEE/OSA Journal of Lightwave Technology*, 19(4):433–455, April 2001.
4. Sudhin Bengeri. Differentiated services support for the Helios optical WDM testbed. Master's thesis, North Carolina State University, http://www.lib.ncsu.edu/etd/public/etd-16201418610131981/etd.pdf, August 2001.
5. E. Hall et al. The Rainbow-II gigabit optical network. *IEEE Journal Selected Areas in Communications*, 14(5):814–823, June 1996.
6. M. Kuznetsov et al. A next-generation optical regional access network. *IEEE Communications*, 38(1):66–72, January 2000.
7. R. E. Wagner et al. MONET: Multiwavelength optical networking. *Journal of Lightwave Technology*, 14(6):1349–1355, June 1996.

8. O. Gerstel, B. Li, A. McGuire, G. N. Rouskas, K. Sivalingam, and Z. Zhang (Eds.). Special issue on protocols and architectures for next generation optical WDM networks. *IEEE Journal Selected Areas in Communications*, 18(10), October 2000.
9. B. Mukherjee. WDM-Based local lightwave networks Part I: Single-hop systems. *IEEE Network*, pages 12–27, May 1992.
10. Zeydy Ortiz, George N. Rouskas, and Harry G. Perros. Scheduling of multicast traffic in tunable-receiver WDM networks with non-negligible tuning latencies. In *Proceedings of SIGCOMM*, pages 301–310, September 1997.
11. George N. Rouskas and Vijay Sivaraman. Packet scheduling in broadcast WDM networks with arbitrary transceiver tuning latencies. *IEEE/ACM Transactions on Networking*, 5(3):359–370, June 1997.
12. Vijay Sivaraman and George N. Rouskas. A reservation protocol for broadcast WDM networks and stability analysis. *Computer Networks*, 32(2):211–277, February 2000.
13. Dhaval Thaker. Multicasting in a partially tunable broadcast WDM network. Master's thesis, North Carolina State University, http://www.lib.ncsu.edu/etd/public/etd-120143410141221/etd.pdf, May 2001.