

TRON: The TOOLKIT for ROUTING in OPTICAL NETWORKS

Ghassen Ben Brahim* Bilal Khan† Abdella Battou‡
Mohsen Guizani § Ghulam Chaudhry ¶

Abstract

The Toolkit for Routing in Optical Networks (TRON) is a freely available library developed to facilitate research experiments on OSPF-based routing protocols for optical networks. Currently, TRON supports the *LightWave-OSPF* routing protocol, which is our adaptation of the optical extensions to OSPF proposed in the internet drafts of Kompella et al. [7] and Wang et al. [9]. TRON is implemented in C++ using the Component Architecture for Simulating Network Objects (CASiNO). TRON software can be used in either simulation or emulation mode. It has permitted us both to simulate *LightWave-OSPF* routing in large optical networks, as well as to emulate routing on a live optical switch. In this paper, we describe *LightWave-OSPF* and the architecture of the TRON software.

1 Introduction

In wavelength division multiplexing (WDM) networks, communication between optical cross-connect (OXC) switches takes place along all-optical WDM channels, commonly referred to as *lightpaths*. A routing protocol for WDM must therefore facilitate two basic tasks: (1) the computation of a path for each pair of source-destination switches wishing to communicate, and (2) the assignment of a wavelength on each optical fiber link along the computed path.

Significant efforts are being made (e.g [2, 4, 7, 9]) to adapt existing routing protocols for use in the WDM domain. In [3], the authors evaluated these efforts and concluded that the OSPF-based proposals being developed offered the greatest potential for success, promising a scalable solution that takes into consideration the goals of Traffic Engineering (TE) and Quality of Service (QoS). Since then, the authors have implemented *LightWave-*

*Electrical & Computer Engineering Department, University of Missouri Columbia.

†Advanced Engineering & Sciences ITT contracted to The Center for Computational Sciences, Naval Research Laboratory, Washington D.C.

‡Princeton Networks Inc, Greenbelt, Maryland.

§Computer Science Department, University of West Florida.

¶Electrical & Computer Engineering Department, University of Missouri Columbia/Kansas City.

OSPF, a routing protocol based on the optical extensions to OSPF proposed in [7] and [9].

The paper is organized as follows: Section 2 provides a brief overview of the OSPF protocol. In section 3, we discuss how *LightWave-OSPF* extends standard OSPF. Section 4 describes how TRON might be integrated with a lightpath provisioning system. Section 5 gives a quick overview of the CASiNO framework, and then describes the implementation of the *LightWave-OSPF* stack and TRON software in terms of CASiNO components. Finally, Section 6 presents our conclusions and describes future work.

2 An Overview of OSPF

When a router boots, lower level protocols determine which of its interfaces are operational. On each operational interface, the router runs the *Hello* protocol to determine the identities of its neighboring routers. On broadcast networks, the Hello protocol is also used to elect the designated router. The designated router is the only local router that transmits and receives routing updates from external networks; all other local routers only transmit to and receive information from their designated router.

After the Hello protocol has determined the list of neighbors on a particular interface, the *Adjacency* finite state machine (FSM) for that interface becomes operational. The protocol implemented by this FSM is responsible for synchronizing the topology database of a router with those of its neighbors.

Each router periodically originates updated link state advertisement (LSA) for each of its interfaces (incident links). OSPF routers distribute their LSA updates by forwarding any changes to their neighbors in a process known as *Flooding*, thereby dynamically maintaining synchronization of their topology databases.

Using its topology database, a router builds a graph representation of the network. Optionally, it may compute the shortest-path tree in this graph (rooted at the router itself) and from this tree compute a next-hop routing table. Externally derived routing information appears within the tree's leaves. The LSA packet format distinguishes between information acquired from external sources and information acquired from internal routers, so there is no ambiguity about the source or reliability of routes. Externally derived routing information (for example, routes learned from EGP or BGP) is passed transparently through the autonomous system and is kept separate from OSPF's internally derived data. Each external route can also be tagged by the advertising router, thereby enabling the passing of additional information between routers at the borders of the autonomous system.

3 LightWave-OSPF

LightWave-OSPF largely follows the internet draft [7] of Kompella et al. which presents

an extension to OSPF for routing in optical networks. In that draft, the authors specify some of the optical network characteristics that must be maintained in an OXC routing database, and show how OSPF’s Opaque¹ LSAs can be used as a vehicle for advertising these parameters within the OSPF protocol. A brief outline of their proposal will be given in the full version of this paper.

Like the earlier drafts of Chaudhuri et al. [4] and Basak et al. [2], Kompella et al. take the stand that an optical adaptation of the OSPF protocol should not advertise any information pertaining to wavelength availability in link state advertisements. They contend that the set of available wavelengths changes so frequently that advertising these changes would not yield a performance increase commensurate to the communication cost of increased control traffic. They recommend postponing the problem of wavelength assignment to the later stage when the lightpaths are being signaled/provisioned. On this point, the present implementation of *LightWave-OSPF* breaks from their proposal. Instead *LightWave-OSPF* adopts the strategy outlined in the internet draft of Wang et al. [9]. There, the authors stipulate that an optical adaptation of the OSPF protocol *should* advertise *both* the number of available wavelengths per fiber *and* the total available bandwidth. To address the objection that advertisement of the available bandwidth is impractical because of rapid changes in network link usage, the authors of the Wang et al. incorporate a tunable threshold to determine when a change is “significant” enough to mandate readvertisement.

Our rationale for advertising wavelength availability information is as follows. In a heterogeneous optical network where a significant fraction of the switches are not wavelength-conversion capable, the absence of wavelength information in the description of network links causes the route computation algorithm to operate with insufficient information. As wavelength utilization of network links increases, the probability of selecting an effective source route that can actually be provisioned dramatically decreases, for although many routes exist from source to destination, wavelength continuity constraint at the transit non-wavelength-conversion-capable switches render most of these computed routes infeasible. We considered it unacceptable that the infeasibility of these routes would not be detected until signaling was attempted (and failed), because this would cause each lightpath setup to experience a large number of “crankbacks”. The cumulative effect of not advertising wavelength availability being that as the load on the network increases, the lightpath setup latency increases prohibitively (even in the absence of contention between concurrent setup requests). This problem gets worse when a larger fraction of switches in the network are non-wavelength-conversion-capable.

LightWave-OSPF minimizes the increase in control traffic due to wavelength advertisement by incorporating the recommendations presented by Apostolopoulos et al. in [1]. Specifically *LightWave-OSPF* implements a range of policies to determine when a router should flood a new LSA advertising a change in its link metric. The policies presently implemented include: (1) *Threshold based policies* that trigger updates when the difference between the previously flooded and the current value of available link bandwidth is larger than a configurable threshold, and (2) *Timer based policies* which generate updates

¹Opaque LSAs were first introduced by Coltun [5] as a way to extend the OSPF protocol.

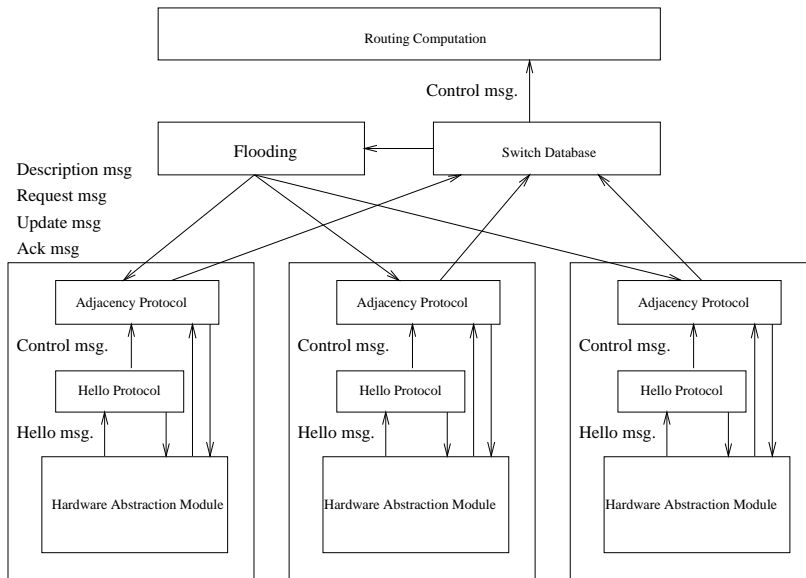


Figure 1: TRON Architecture

at fixed time intervals, or enforce a minimum spacing between two consecutive updates.

3.1 TRON Architecture

The TRON architecture can be decomposed functionally into a set of cooperating modules, as shown in figure 1. The modular design facilitates future modifications and extensions to the TRON software. Because each OXC switch has several interfaces, certain modules such as the “Adjacency Protocol”, the “Hello Protocol”, the “Network Interface”, and the “Hardware Abstraction” modules have to be reproduced for each interface. In contrast, the “Routing”, “Flooding” and “Switch Database” modules are instantiated once for each OXC switch. We will briefly description each of these components in the next section.

1. Network Interface and Hardware Abstraction modules

The TRON stack is built over these two modules. Together, they are responsible for reporting any failure or change in local switch resources, by generating events and forwarding them upwards for the rest of the TRON stack to respond to.

The Network Interface module represents the connection between the OXC router and the network. This module informs the router whether the interface to the physical network is up or down, and to maintain information about the type of network to which the interface attaches, the interface’s IP address, the area ID, a list of neighboring routers, the router priority, the IP address of the Designated

Router (DR) and Backup Designated Router (BDR).

The Hardware Abstraction module monitors the OXC for alarms and alerts triggered by the failures or changes to the OXC's internal components. The Hardware Abstraction module maps these physical hardware dependent signals to a set of generic messages that are processed by the Hello and Flooding module. An example of generic messages are the `OXCPortUp` and the `OXCPortDown` messages which indicate a port's operation and failure, respectively. Extending TRON to work with other optical switches then requires only modification to the Hardware Abstraction module.

2. Hello module

The Hello module implements the Hello protocol, which is responsible for maintaining neighbor relationships between OXCs. Specifically, the Hello protocol serves to certify that the links connecting neighboring routers correctly support bidirectional communication. The protocol specifies that every `HelloInterval` seconds, each router sends a Hello packet containing the identities of the neighbors that it knows about. `TwoWay` connectivity is attained when the local router's ID is listed in the Hello packet received from a remote router; otherwise, the connectivity is declared to be only `OneWay`. The Hello protocol is also responsible for electing a Designated and Backup Designated Routers, by carrying election priority information in the Hello packets.

The Hello module is located directly above the Network Interface and Hardware Abstraction modules, and the Hello protocol is sensitive to messages generated by these modules.

3. Adjacency module

The Adjacency module lies above the Hello module within the TRON stack, and implements the Adjacency/Flooding (A/F) protocol. The A/F FSM (see figure 2) is quite similar to the adjacency FSM of the original OSPF. The A/F FSM is triggered by the Hello protocol: Upon reaching the `TwoWay` state the Hello protocol sends a `Begin` control message to the A/F FSM triggering it to become operational by entering the `Extstart` state.

The Adjacency protocol is responsible for the synchronization of Switch Databases of neighboring routers. The exchange protocol is asymmetric: the first step is to determine a "master" and a "slave" relationship. After the master-slave relationship negotiation, the new state is `Exchanging`. The two routers then exchange Description packets containing a summary of their respective databases which lists the advertisements within it. Based on this, the routers will request and exchange the information as necessary to synchronize their databases. This is achieved using Request packets, Update packets, and Acknowledgment packets. The routers use checksums to ensure that all protocol packets have valid contents. Once all necessary exchanges have been completed, the protocol enters the `Full` state and the Flooding module is notified. Subsequently, the Adjacency module is subservient to the Flooding module which may periodically give it LSAs to flood. The state transition diagram of the A/F FSM is depicted in figure 2

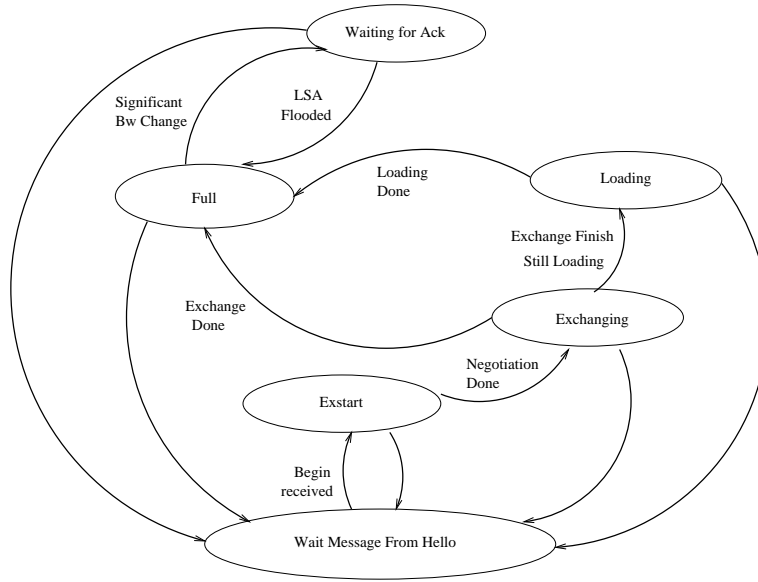


Figure 2: A/F FSM for *LightWave-OSPF*

The Adjacency protocol is terminated if the Hello module sends a message up withdrawing certification of `TwoWay` connectivity with the peer. Such failures are detected by the Hardware Abstraction and Network Interface modules, which inform the Hello protocol which in turn notifies the A.F FSM by sending a `halt` control message. The latter message causes the interruption of the A/F FSM and its transition to the initial state `WaitMessageFromHello`.

In *LightWave-OSPF*, the Update packets carry a list of Link State Advertisements (LSAs). These LSAs are encapsulated into Opaque LSAs. The Opaque LSAs carry different types of Type/Length/Value (TLVs) and sub-TLV packets. The TLVs and sub-TLVs include the following: Traffic Engineering Metric, Maximum Bandwidth, Unreserved Bandwidth, Maximum Reservable Bandwidth, Resource Class/Color, Link Media Type, Shared Risk Link Group, Available Wavelength, Number of Active as well as Preemptable Channels.

4. Flooding module

The Flooding module may interact with all instances of the Adjacency modules. This interaction begins only after the Adjacency module has fully synchronized with the remote peer. The Flooding module is invoked whenever one of the following three events occurs:

- (a) When there is a failure in one of the incident links as detected by the Hello module. In this scenario, the Flooding module must flush the LSA for the failed link from the network. It does this by inserting a new LSA for the link into its Switch Database (with a higher sequence number and age set to

MaxAge) and forwarding it out on all interfaces, via their respective Adjacency modules.

- (b) In case of a significant change in the available bandwidth of an incident link. In this scenario, the Flooding module inserts a new, more accurate LSA describing the link into its Switch Database—with a higher sequence number—and forwarding the new LSA out on all interfaces, via their respective Adjacency modules.
- (c) When new LSA arrives with more up-to-date information about the network's topology/attributes. In this scenario, the Flooding module inserts the newer LSA into the Switch Database, acknowledges the receipt of the update, and then forwards the new LSA on all interfaces via their Adjacency modules *with the exclusion of the Adjacency module that reported the update to the Flooding module*.

5. **Switch Database** module

This module maintains a collection of LSAs which have been collected by the Adjacency protocol and subsequent operation of the Flooding module. Upon the receipt of a new LSA, the SwitchDatabase determines whether the LSA is a newer version of an LSAs that it already owns. If so, the newer LSA replaces the older one; otherwise the LSA is discarded. The SwitchDatabase is also responsible for continuously aging the LSAs and removing those whose age reaches **MaxAge**. The Switch Database also periodically reoriginates LSAs (with higher sequence number and age set to zero) that have been previously generated by the router, thereby ensuring that they do not disappear from the Databases of other routers in the network.

6. **Routing** module

The Routing module has the main task of building a graph representation of the network based on the contents of the Switch Database module, and responds to requests for the computation of routes to destination routers. This route computation may require consideration of Traffic Engineering and QoS requirements for the connection, and reconciling them with the corresponding information within the LSAs. Traditionally, the next-hop along these routes is cached in a routing table for hop-by-hop routing. In the case of *LightWave-OSPF*, however, the entire route is used as a source-route; the lightpath provisioning protocol (e.g. the provisioning protocols of CHIME-NET [6]) attempts to provision the path along the specified route.

4 Integration of Provisioning/Signaling with TRON

The TRON software can be considered as the routing computation component of the provisioning protocols, which are in turn viewed as clients of TRON. These clients use the information available computed by TRON to physically setup lightpaths. In our present scheme, the lightpath is determined by source routing establishment, unlike hop-by-hop schemes such LSP establishment in MPLS.

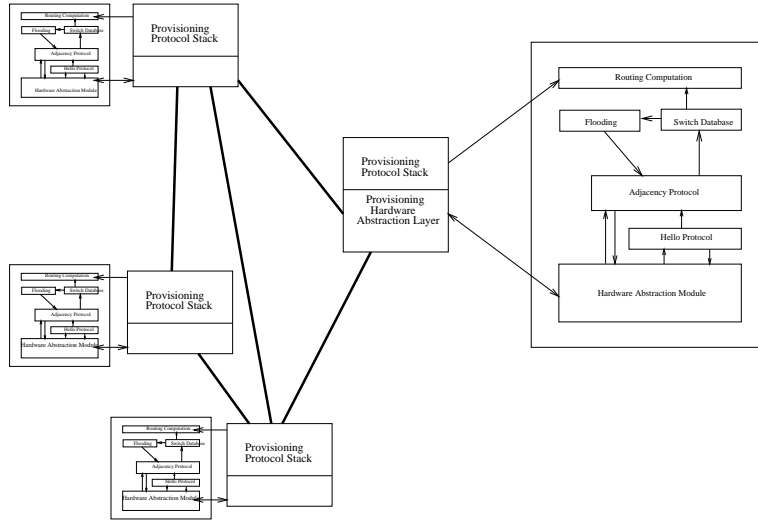


Figure 3: Interaction of Optical OSPF and Provisioning protocols

Once the provisioning protocol gets the necessary information from the Routing module, it starts physically setting up the path by establishing bindings at each of the transit switches. If it succeeds, the provisioning protocol informs the Hardware Abstraction modules of every switch along the route about the change in bandwidth availability on affected links. The Hardware Abstraction modules report this change, and if the change is determined to be significant, a new link state advertisement of the affected link is reoriginated and flooded to all neighbor routers. If the change is not significant, then the notification is ignored. Figure 3 represents the interaction between TRON and the provisioning protocol. We are in the process of integrating TRON with lightpath provisioning in CHIME-NET, our distributed optical network management system [6].

5 Implementation

TRON is implemented using the Component Architecture for Simulating Network Objects (CASiNO), a framework for rapid prototyping of communication protocol stacks and network simulators.

5.1 The CASiNO Framework

CASiNO is a visualizable C++ framework [8] that implements a rich, modular coarse-grained dataflow architecture, with an interface to a reactor kernel that manages the application's handlers for asynchronous I/O, real timers, and custom interrupts. A brief description of the CASiNO framework components will be provided in the full version of

this paper.

5.2 Implementing TRON using CASiNO

In the full version of this paper, we will give a brief description of TRON's implementation in terms of CASiNO framework components.

6 Conclusion and Future Work

TRON implements the *LightWave-OSPF* routing protocol, which is an adaptation of the optical extensions to OSPF proposed in [7] and [9]. The Toolkit for Routing in Optical Networks (TRON) is a simulation/emulation library developed to facilitate experiments on OSPF-based routing protocols for optical networks. In this paper, we described *LightWave-OSPF* and the architecture of TRON. We hope that TRON will serve as a starting point for other research work on OSPF-like protocols for optical networks.

In simulation, TRON is being used to investigate the performance of *LightWave-OSPF* routing in large optical networks. In particular, these experiments seek to give a quantitative cost-benefit evaluation of advertising wavelength availability under various network environment assumptions. In emulation, TRON is being used to measure routing performance between the WSXC switches within the Multi-wavelength Optical Networking (MONET²) Program. When complete, these investigations and experiments will be the subject of a future publication.

We are also in the process of integrating TRON with lightpath provisioning support in CHIME-NET, a distributed network management system which provides connection and fault management for wavelength division multiplexing networks [6]. Eventually, TRON will provide dynamic topology discovery for CHIME-NET.

Presently, when running in emulation mode, TRON sends all its control messages out-of-band, on an auxiliary control network operating IP over ATM. Each optical switch running a TRON stack must therefore be configured with information about the identities of its neighboring optical switches. Specifically, it must be told the location and listening port of the TRON stacks of each of its neighbors, so that it can establish out-of-band connectivity to these peers prior to booting. In future releases of TRON, we hope to consider in-band transport of routing control messages using a dedicated control wavelength.

References

- [1] G. Apostolopoulos, R. Guerin, S. Kamat, and S. Tripathi. Quality of Service Based Routing: A performance Perspective. Proceedings of SIGCOMM, September, 1998.

²MONET is sponsored by the Defense Research Project Agency

- [2] D. Basak, D. Awduche, J. Drake, and Y. Rekhter. Multi-Protocol Lambda Switching: Issues in Combining MPLS Traffic Engineering Control with Optical Corssconnects. Internet Draft, January, 2000.
- [3] G. B. Brahim, B. Khan, A. Battou, M. Guizani, and G. Chaudhry. Routing for Optical Networks. Parallel and Distributed Computing and Systems, November 2000.
- [4] S. Chaudhuri, G. Hjalmtysson, and J. Yates. Control of Lightpaths in an Optical Network. Internet Draft, February, 2000.
- [5] R. Coltun. The OSPF Opaque LSA Option. Request for Comments No. 2370, July, 1998.
- [6] B. Khan, D. Kleiner, and D. Talmage. CHIME-NET: Towards a distributed optical network management system. submitted to the 2001 IEEE Workshop on High Performance Switching and Routing, Dallas, Texas, USA, November 2000.
- [7] K. Kompella, Y. Rekhter, D. Awduche, J. L. G. Hjalmtysson, S. Okamoto, D. Basak, G. Bernstein, J. Drake, N. Margalit, and E. Stern. Extensions to IS-IS/OSPF and RSVP in support of MPL(ambda)S. Internet Draft, October, 1999.
- [8] S. Mouncastle, D. Talmage, S. Marsh, B. Khan, A. Battou, and D. C. Lee. CASiNO: A component architecture for simulating network objects. Proceedings of 1999 Symposium on Performance Evaluation of Computer and Telecommunication Systems, (Chicago, IL), pp. 261–272, Society for Computer Simulation International, July 1999.
- [9] G. Wang, D. Fdyk, V. Sharma, K. Owens, G. Ash, M. Krishnaswamy, Y. Cao, M. Girish, H. Ruck, S. Bernstein, P. Nquyen, S. Ahluwalia, L. Wang, A. Doria, and H. Hummel. Extensions to OSPF/IS-IS for Optical Routing. Internet Draft, March, 2000.