

Simple Wavelength Assignment Protocol

Stephen Suryaputra
ssuryapu@nortelnetworks.com
Nortel Networks*

Joseph D. Touch, Joseph Bannister
{touch, joseph}@isi.edu
USC/Information Sciences Institute

October 26, 1999

ABSTRACT

Coupling IP routers with wavelength-selective optical cross-connects supports existing Internet infrastructure in a wavelength division multiplexing (WDM) optical network. Because optical wavelength routing is transparent to IP, packets can bypass traditional forwarding and route directly through the optical cross-connect, resulting in very high throughput and low delay. This approach shares features with label switching, but wavelengths are a much more scarce resource than labels. Because optical switches have larger switching times than electronic switches, and wavelength conversion are expensive, wavelength "label" swapping is not easily done. Wavelength "label" assignments must consider these limitations to be practical in an optical environment. This paper describes an optical label switching signaling protocol, SWAP, which takes into account the optical device limitations, and is designed to minimize wavelength swappings, utilize wavelengths with merging of flows and reduce the reconfiguration of optical switches.

INTRODUCTION

Coupling Internet protocol (IP) routers with wavelength-selective optical cross-connects enables existing Internet infrastructure to operate in a wavelength division multiplexing (WDM) optical

network. Because optical wavelength routing is transparent to IP, very high throughput and low delay can be achieved when packets bypass the IP forwarding process by being routed directly through the optical cross-connect. While this approach is similar to label switching in general, there are several issues limiting its feasibility. Current label switching mechanisms assume a large label space, where label swapping is inexpensive (done in hardware at the ATM layer). Replacing labels with colors (wavelengths) in a WDM network raises several challenges. Current WDM technology is limited to a few (eight to 64) wavelengths per link, which is very small compared to the number of fine-grain network connections in the Internet today [NLANR]. Wavelength converters (corrolary to label swapping) are expensive [Yoo96]. Finally, practical active (data-dependent) optical switching elements are slow, and thus costly to reconfigure. Of the four types of optical switching elements: mechanical, electro-optical, thermo-optical, and SOA-based gate switched, only electro-optical has low switching time, but it suffers from high crosstalk and loss [RaSi98]. Thus reconfiguration of a practical optical wavelength cross-connect is generally slow.

These limitations warrant re-examination of IP-WDM switching approaches. The performance of an instance of this approach, called Packet over Wavelengths (POW) has been simulated and studied [BTWS99]. This study is based on a signaling protocol created to dynamically configure the lightpath for flows. This protocol is called the

* Work performed while at USC/ISI

Simple Wavelength Assignment Protocol (SWAP). SWAP modeling and simulation is performed on the VINT/ns simulator [NS].

PRIOR AND RELATED WORK

POW forwarding is similar to label switching, i.e., they both replace complicated IP forwarding processing with a comparatively simple label lookup. IP processing relies on longest-prefix match in a large routing table, followed by updates of portions of the IP header, such as hopcount (TTL) and checksum. Label processing indexes the label in a small, fixed table, and performs a fixed label substitution. In POW, that lookup is done in the physical layer instead of in the link or network layer by the wavelength of the packet signal. This section includes further discussion of label switching approaches because of this similarity.

There are at least five known approaches to label switching: Toshiba Cell Switch Router (CSR), Ipsilon IP Switching, Cisco Tag Switching, IBM Aggregate Route-Based IP Switching (ARIS), and Multiprotocol Label Switching [DaDR98]. Among these, most notable are Tag Switching and IP Switching. Although MPLS is the IETF standard for label switching, its principal operations are significantly influenced by Tag Switching; this discussion thus regards these two approaches as the same. The primary difference between Tag Switching and IP Switching is the process of assigning labels to IP flows. Tag switching assigns flows based on routing protocol events; IP switching is based on the actual traffic seen on the network. The taxonomy described in [DaDR98] thus considers Tag Switching as control-driven and IP switching as data-driven.

POW avoids routing protocol interactions, making it more responsive to traffic trends. As a result, it is based on a data-driven approach similar to that of IP switching. Although IP switching has not performed well in real systems due to an overabundance of switchable flows, POW takes further steps, including coarsely aggregating flows to and merging flows, to reduce the number of flows handled by each switch. By merging, POW localizes the traffic dynamics to the nodes near the edges of the network.

DESIGN

The POW Switch Architecture consists of four major components (Figure 1):

- Control Processor: an IP router equipped with gigabit data interface(s) and a control interface. Depending on the scalability-complexity trade-off, multiple data interfaces of the control processor could be implemented as a single electronic interface, where the optical path is terminated early but the optical interface ID must be stored in the form of internal headers (increased complexity).
- Optical Fabric: an interconnection network of optical switching elements. The elements are capable of switching optical signals from one input port to one of a set of output ports, where the output port is selected based on the wavelength of the signal at the input port.
- Wavelength Converters: these devices convert input signals with wavelength λ_{in} to output signals with wavelength λ_{out} .
- Wavelength MUX/DMUX: redirect a specific wavelength to a particular fabric, and merge the output of the fabrics to the external fibers. These are special cases of statically configured optical switching elements.

Although the details of the components are beyond the scope of this paper, the arrangement of the components follows *share-with-local wavelength-convertible* switch architecture [LeLi93] to reduce the number of wavelength converters per node.

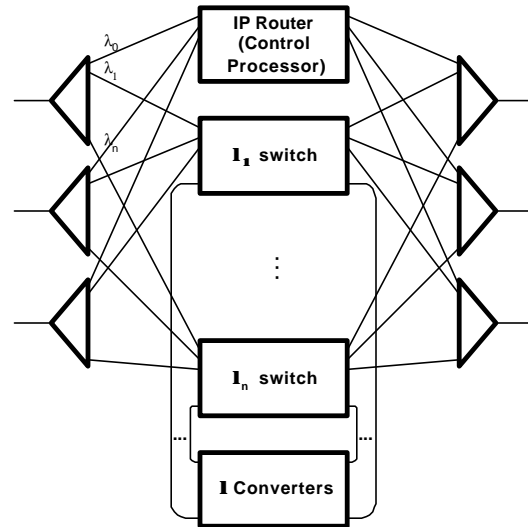


Figure 1 - POW Switch Architecture

There are several high-level requirements for the POW signaling protocol. First, it must be kept as simple and as light (using few, brief messages) as possible. Second, the signaling should construct the

continuous light path as far as possible. This second requirement is driven by the hardware limitation. Non-continuous light paths require wavelength conversion, which requires expensive hardware (this its use should be limited). Third, the protocol must support flow merging (grooming). These high-level requirements are the motivation for the following design decisions for Simple Wavelength Assignment Protocol (SWAP) signaling protocol¹.

To conform with the first requirement, i.e.: simplicity, SWAP is implemented on top of a reliable transport layer; this decouples the protocol from its reliable transmission. This is common practice for signaling protocols, e.g. the ATM signaling protocol is implemented on top of Service-Specific Connection-Oriented Protocol (SSCOP) which is a reliable transport above AAL5 [Kesh97]. The telephone Signaling System 7 (SS7) makes the same choice. SWAP components establishes per-neighbor TCP connections, over which the signals are sent. Neighbor connections enable the switch to determine whether it is the first, the last or the intermediate hop for a particular flow, which is used to simplify the signaling, failure detection, and recovery. Neighbor relationships are maintained per link of the switches (one connection per link). The neighbor connections are not optimized because they are trivial and similar to any of the common routing protocols (IS-IS, OSPF, or BGP) [Perl92].

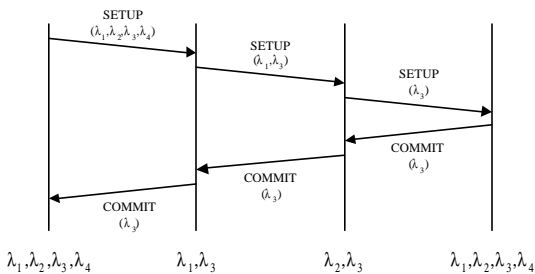


Figure 2 - Time sequence diagram of first-hop-initiated signaling

Constructing the contiguous light path as far as possible requires SWAP to pick a common free wavelength along the flow path, therefore SWAP

¹ SWAP is called the Flow Management Protocol (FMP) in the earlier POW paper [BTWS99]. FMP was renamed SWAP to differentiate it from Ipsilon’s Flow Management Protocol (IFMP).

must collect the list of free wavelengths for each hop. If there is one free wavelength common to all the hops, it will be picked. If not, SWAP may choose to construct a non-contiguous light path (if so configured) if there are sufficient wavelength converters available. This feature is unique to SWAP compared to label swapping techniques, because SWAP’s decision to pick a “label” (wavelength here functions as label) is a global decision. SWAP tries to minimize or eliminate swapping of “labels”. Therefore, SWAP incurs a round-trip time to select an wavelength. The resource must be also locked during signaling.

Next SWAP decides where to initiate the signalling. Either end of the path is appropriate, as they natural places where SWAP can efficiently gather complete path information. The first hop is good because a source SWAP can propagate its free wavelength set when it detects an active flow (SETUP). When the next hop receives that set, it intersects that set with its own free wavelength set, and forwards the result to the next hop. Assuming the final set is not empty, the last hop picks one free wavelength from the resulting set, configures its local switch, and send an acknowledgement (COMMIT) back to the previous hop with the chosen wavelength. Upon receiving an acknowledgement, the previous hop configures its local switch and passes the acknowledgement to its previous hop, until the packet is received by the first hop. The process is illustrated in Figure 2².

However, it is better to initiate the signaling from the last hop, for a number of reasons, notably in the presence of grooming (merging). In merging, there is a single last hop, but multiple first hops, which would complicate a source-initiated protocol. The last hop will also notice the flow earlier, as the traffic merges there. Second, it will simplify the protocol because there could be more than one outstanding setup request from upstream and the protocol must keep track of the upstream status so that it can selectively send the COMMITs back to it. Third, last-hop-initiated signaling will ease the interaction with RSVP [ZDE+93] in the future. ARIS takes the same approach as it allows route aggregation.

The drawback of last hop initiated signaling is that it will take more time to complete the signaling. The first sequence is similar to the first-hop initiated,

² λ_n is the free wavelength set of that hop.

however the previous hop should not start sending the packets using the new wavelength unless the next hop has already setup the switch (to avoid losses). SWAP could do something similar to IP Switching, that the switch will send the packets using the slow path (through the IP router) while waiting for response from the next hop. However, because this technique requires temporary path termination and optical switches take a long time to setup, it is undesirable to do so. Instead in SWAP, the first hop will wait one round-time for signaling propagation to complete. The process is shown in Figure 3³.

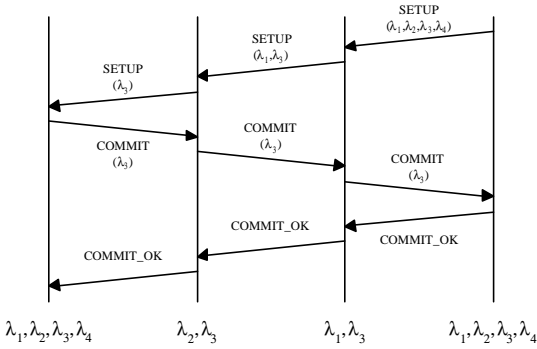


Figure 3 – Time sequence diagram of last-hop-initiated signaling

Flow aggregation (grooming) also affects where to initiate the teardown mechanism. The last hop is undesirable, because drops in an aggregated flow are noticed at the sources first. Merging hops is also not desired because it requires the hop to monitor the optical signal. Therefore, it is the responsibility for the first hops to initiate the teardown. If the first hop switches of a switched flow detect a drop in the throughput, it will send TEARDOWN to the next hop and the next hop will pass it to further hops if there are no switched incoming branches. Figure 4 illustrates this effect on an aggregated flow.

³ There are common free wavelengths for all examples in this section. Wavelength conversion will be described later in implementation section.

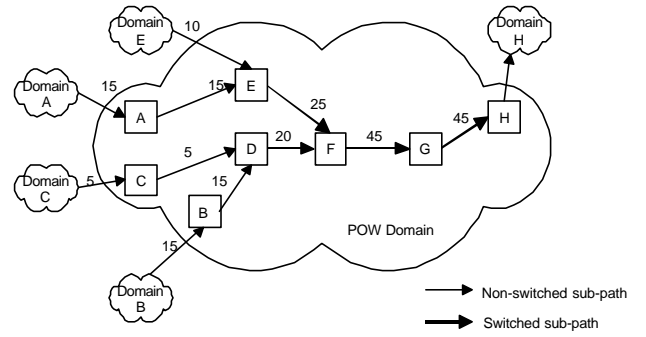


Figure 4 – Flow Aggregation Scheme

Switch	Link	Free Wavelengths Set
H	GH	$\mathbf{I}_{GH} = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$
G	FG	$\mathbf{I}_{FG} = \{\lambda_2, \lambda_3, \lambda_4\}$
F	EF	$\mathbf{I}_{EF} = \{\lambda_3, \lambda_4\}$
	DF	$\mathbf{I}_{DF} = \{\lambda_1, \lambda_2\}$
D	CD	$\mathbf{I}_{CD} = \{\lambda_1, \lambda_2, \lambda_4\}$
	BD	$\mathbf{I}_{DF} = \{\lambda_3, \lambda_4\}$

Table 1 – Free wavelengths at Figure 4 switches

Suppose SWAP was set to regard 20 pps⁴ as a threshold to switch a flow. Switch D, F, G, and H all see an aggregate outgoing throughput of 20 pps or higher for flow F_1 (* → Domain H, i.e., traffic going to H) and because switch H knows it is the last hop, it locks the free wavelength resource \mathbf{I}_{GH} and send $SETUP(F_1, \mathbf{I}_{GH})$ to G. Upon receiving that $SETUP$, G does the wavelength set intersection $\mathbf{I}_x = \mathbf{I}_{GH} \cap \mathbf{I}_{FG} = \{\lambda_2, \lambda_3, \lambda_4\}$, lock \mathbf{I}_{FG} and sends $SETUP(F_1, \mathbf{I}_x)$ to F. F does the same intersection $\mathbf{I}_y = \mathbf{I}_x \cap \mathbf{I}_{DF} = \{\lambda_2\}$ and $\mathbf{I}_z = \mathbf{I}_x \cap \mathbf{I}_{EF} = \{\lambda_3, \lambda_4\}$. Then it will send $SETUP(F_1, \mathbf{I}_y)$ to D and $SETUP(F_1, \mathbf{I}_z)$ to E. Both D and E know that they are the first hops for that path⁵ and D will send $COMMIT(F_1, \lambda_2)$ and E will send $COMMIT(F_1, \lambda_3)$. Meanwhile, F waits for responses from both. Upon receiving the responses, F picks λ_3 because it is an element of \mathbf{I}_{EF} and E contributed more to the aggregate throughput than D. F removes λ_3 from the set \mathbf{I}_{EF} and λ_2 from the set \mathbf{I}_{DF} , and

⁴ SWAP uses an X/Y classifier [LiMc97].

⁵ The switches will use two criteria to determine whether they are the first hop: there is no upstream neighbor for the path, or there is no incoming branches that have a high incoming throughput.

sends $\text{COMMIT}(F_1, \lambda_3)$ to G. G removes λ_3 from the set \mathbf{I}_{FG} , unlocks it and forwards $\text{COMMIT}(F_1, \lambda_3)$ to H. Then, H removes λ_3 from the set \mathbf{I}_{GH} , unlocks it, configuring its optical switch and flow converter to convert λ_3 back to the default wavelength λ_0 , and send $\text{COMMIT_OK}(F_1)$ to G. Upon receiving $\text{COMMIT_OK}(F_1)$ from H, G sends $\text{COMMIT_OK}(F_1)$ to F, and switch F starts sending the flow through a wavelength converter that converts default wavelength λ_0 to λ_3 .

Branch E is selected because the more a branch contributes to the aggregate throughput, the more likely it is expected to stay significant or become even more significant. If other branches become inactive, they will likely become a sub-path. The wavelengths of low flow branches are picked arbitrarily and merged to the target wavelength.

Now suppose there is an increase in the F_1 throughput from B to D. D realizes that the flow has been switched, so it sends $\text{SETUP}(F_1, \mathbf{I}_{BD})$ to B⁶. B determines that it is the first hop and sends $\text{COMMIT}(F_1, \lambda_3)$. Upon receiving from B, D configures its optical switch and sends $\text{COMMIT_OK}(F_1)$ to B, and B sends the flow using λ_3 when it receives the message. If, subsequently, E detects that F_1 throughput drop, it will send $\text{TEARDOWN}(F_1)$ to F. Switch F will not forward the message further because it still has a switched branch, i.e.: DF. As a result, F only frees the wavelength, removes the switched-path from its optical switch to its IP router, and its IP router will merge the incoming flow to the outgoing switched flow.

Finally, SWAP requires that neighbor protocol emit periodic keep-alive messages so the POW switch will detect neighbor failures. The keep-alive message should incorporate a mechanism to detect the case of neighbor failure and subsequent recovery and up again prior to the timeout of its neighbor entry. This is handled by the routing protocol. If the failed neighbor is the previous hop, then the switch will do the same thing as if it received $\text{TEARDOWN}(F_1)$, $\text{TEARDOWN}(F_2)$, ..., $\text{TEARDOWN}(F_n)$ where F_{1-n} are the switched flows coming from the neighbor. If the failed neighbor is

⁶ There is no attempt to maintain wavelength continuity after the lightpath has been set up. However, if λ_2 is an element of \mathbf{I}_{BD} then D will send $\text{SETUP}(F, \{\lambda_2\})$.

the next hop and there is no previous hop switched, then the switch just sends the flow using the default wavelength λ_0 . However, if there is a switched previous hop, the switch will take the last hop role, i.e., converting the switched flow back to λ_0 .

To summarize the SWAP design points:

1. Implemented on top of a reliable transport protocol.
2. First-hop is defined as the point where there is no upstream neighbor, or there are upstream neighbors but the incoming throughputs are not high enough. Last hop is defined as the point where there is no downstream neighbor.
3. Use last-hop-initiated setup if there is no switched path and aggregation-point-initiated setup if the aggregation point already has the flow switched (i.e., the aggregation point is the last hop of the augmentation to the existing switched path)
4. Use first-hop-initiated teardown. Teardown messages are terminated at the merging point if there is still a switched incoming branch.
5. Resources (free wavelengths) are maintained and locked independently for each incoming link.
6. Grooming points should maintain the flow status for each incoming branch and the flow packet count for each non-switched incoming branch.
7. Whether the switch is the first, the intermediate or the last hop is determined using the neighbor protocol.
8. The neighbor protocol is assumed to be available or be provided by routing protocols.
9. If the neighbor protocol reports that there are no upstream and downstream neighbors, SWAP will be disabled because SWAP implementation assumes there are at least two POW switches in sequence (actually, it is useful only if there are at least three switches in a row).

IMPLEMENTATION

There are 6 (six) messages related to wavelength assignment (connection setup and teardown):

- $\text{SETUP}(F, \mathbf{I})$. Ask the previous hop to pick a wavelength to setup the connection by sending the free wavelength set for the sub-path (also lock it). The recipients intersect this set with their own free wavelength set, and forward the result to the subsequent hop when they detect a high incoming throughput for a flow.

- **SETUP_CONFLICT(F)**. Inform the next hop that one of the wavelength sets along the path has been locked by others. Intermediate hops will pass this message to the next hop. The initiating hop will perform a “backoff” procedure.
- **SETUP_FAIL(F)**. Inform the next hop that the connection cannot be made as there are no free resources⁷. Intermediate hops will pass this message to the next hop if they are not the initiating hop.
- **COMMIT(F, λ)**. Inform the next hop to use λ as the incoming wavelength. Intermediate hops will pass this message to the next hop if they are not the initiating hop.
- **COMMIT_OK(F)**. Inform the previous hop that the optical switch has been setup. Intermediate hops will pass this message to the previous hop. This message will not be forwarded further if there are no branches waiting for it.
- **TEARDOWN(F)**. Inform the next hop to tear down the connection. Intermediate hops will pass this message to the next hop.

In addition to the message types above, SWAP uses an X/Y classifier to detect a flow with the following default parameters:

- FlowDetectionTimer = 20 secs
- FlowActiveTimer = 20 secs
- HighThreshold = 10 packets
- LowThreshold = 5 packets
- BackoffPeriod = 1 ms
- BackoffLimit = 10 retries
- WavelengthConvertEnable = true
for merging, false otherwise
- PartialPathAllowed = false

The messages and the parameters will initiate various state transitions. Each SWAP entity maintains two types of states for each flow entry: incoming and outgoing states. Each incoming branch has its own incoming state, but there is only one outgoing state. Some transitions in the incoming state will initiate further transitions in the outgoing one, and vice versa. This interaction is illustrated in Figure 5 and Figure 6. In these figures, identity (self-loop) transitions are not shown unless the transition causes a message to be sent. Such transitions are indicated by a transition number inside the state, as in case of transition O4 in Figure 6.

To understand SWAP state transitions, some definitions are provided for the further discussion.

⁷ Resources here could be wavelengths or devices needed to setup the lightpath.

First, a *switchable branch* is a branch with PacketCount higher than HighThreshold and has an upstream neighbor. Second, consecutive switchable branches will potentially construct a lightpath. SWAP could construct several combination of lightpaths: contiguous or non-contiguous, and partial or full. A *contiguous lightpath* is a lightpath that consist of several branches switched using the same wavelength. A *non-contiguous lightpath* is a lightpath that consists of several branches switched using different wavelengths. Non-contiguous lightpaths can be constructed if the SWAP parameter of WavelengthConvertEnable is true⁸. A *partial lightpath* is a path where not all the switchable branches are switched. A partial lightpath is constructed only if SWAP cannot acquire the necessary free resources along the entire path and the parameter PartialPathAllowed is True. *Full lightpath* is a path where all switchable branches are switched.

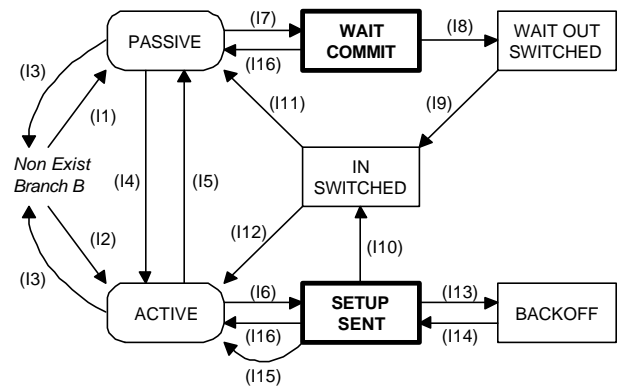


Figure 5 – Incoming State Transition⁹

The state of a traffic flow start as a new incoming flow F is received on a branch B (an interface of the switch). For a flow-branch tuple (F, B), a SWAP process for that particular flow, S, will perform different roles based on the following criterias:

IC1: The output state of F has already been switched.

IC2: S has a downstream neighbor in the direction of F.

In the case of **IC1** is true, then S will actively monitor the throughput of the flow (path I2 in Figure 5). If **IC1** is not true, **IC2** will be used to make the

⁸ This is commonly known as the *wavelength-continuity constraint* [Mukh97]

⁹ Detailed transition triggers and actions are listed in the appendix.

decision. If **IC2** is true, indicating *S* is not the egress node, then *S* creates a state for *F*, passively monitors *F*, and waits for further signaling messages (path I2 in Figure 5). However, if **IC2** is not true, *S* is the egress node therefore it will actively monitor the flow *F*. Both ‘active *S*’ and ‘passive *S*’ need to monitor the flow so that the state for *F* eventually dies with the flow. A timer `FlowDetectionTimer` is used to facilitate this.

Once ‘active *S*’ (*S_A*) determines that the throughput of *F*₁ on branch *B* is high, (*F*₁, *B*) becomes a switchable branch, and *S* will lock **I_B** and send `SETUP(F1, IB)` to its upstream neighbor, *S_P*. If **I_B** was locked by another on-going signaling for flow *F*₂, *S_A* will do a backoff procedure similar to that of CSMA/CD (path I13 and I14 on Figure 5). Once *S_A* completes its backoff, it will retry the `SETUP` procedure. If after `BackoffLimit` tries, *S_A* still cannot get the lock, it will give up the opportunity to switch *F*₁ and monitor it for another detection window.

The upstream neighbor *S_P* (passive role) upon receiving `SETUP(F, Iin)` will use the following criteria:

- IC3:** There is at least one switchable (*F*, *B*).
- IC4:** Full lightpath construction is feasible, which means wavelength continuity is achievable or wavelength conversion is allowed and possible.
- IC5:** Partial lightpath construction is allowed and feasible.

If **IC3** and **IC4** are true, then for every switchable (*F*, *B_n*), *n*=0,1,2,...,N, *S_P* will spawn a task *S_{Pn}*, which locks **I_n**, assign **I** = **I_{in}** ∩ **I_n** and send `SETUP(F, I)` to the upstream neighbor. Here the outgoing state of *F* becomes `WAIT_ALL_COMMIT` (Figure 6). Alternately, if **IC3** and **IC5** are true, *S_P* will pick $\lambda_{out} \in \mathbf{I}_n$ and send `COMMIT(F, λ_{out})` and upon receiving `COMMIT_OK(F)`, it will take the active role (transition O5 and O6 in Figure 6, transition I4 in Figure 5).

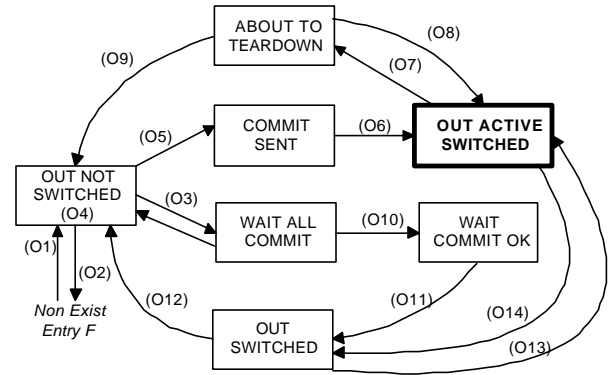


Figure 6 - Outgoing State Transition

On the second phase of signaling, where *S_P* is in the `WAIT_ALL_COMMIT` state, it will collect all `COMMIT(F, λ)` received on any switchable branch *B_n*, *n*=0,1,2,...,N and pick one wavelength λ_x , *x*=0,1,2,...,N such that *B_x* is the branch with the highest throughput. *S_P* then will send `COMMIT(F, λ_x)` downstream. Meanwhile, *S_P* also records the wavelength of preference of each branch¹⁰.

This `COMMIT(F, λ_x)` will eventually be received by *S_A*, the egress node. *S_A* will configure the switch immediately, rather than (as *S_P*) waiting for all `COMMIT` messages to arrive before acting. The *S_A* switch thus converts λ_x to λ_0 (the slow path wavelength), terminating the lightpath.

If the criteria **IC3**, **IC4**, and **IC5** cannot be concurrently met, *S_{Pn}* will send a `SETUP_FAIL(F)` message downstream. This message will not be propagated by the downstream nodes if there are other signaling tasks persisting for a switchable branch *B_m*.

However, if **IC3**, **IC4**, and **IC5** can all be met, but the wavelength sets for the branch were locked, *S_{Pn}* will send `SETUP_CONFLICT(F)` toward *S_A* so that *S_A* can commence its backoff procedure. Setup conflict messages must be propagated along the entire signaling path because every *S_h* along the path has locked the wavelength set for the branch. By releasing those locks, *SWAP* creates opportunities to setup paths belonging to other flows.

¹⁰ *SWAP* could be modified so that λ in `COMMIT(F, λ)` is a set, then wavelength continuity will be improved.

The last part of SWAP signaling handles the teardown mechanism (transitions O7, O8, and O9 in Figure 6). If an active node S detects that the throughput of a switched branch (F, B) drops below `LowThreshold` for a specific detection duration `FlowActiveTimer`, S will monitor that flow for another `FlowActiveTimer` window (transition O7). If after the second window, the throughput remains below `LowThreshold`, S will send a `TEARDOWN(F)` message upstream (transition O8). S then becomes active or passive based on its position in flow tree.

MODELING AND SIMULATION

SWAP was designed to enable the evaluation of the performance of the POW architecture. To ease SWAP development and verification and to evaluate specific POW topology, SWAP was implemented in VINT/ns, a popular network simulation package [NS].

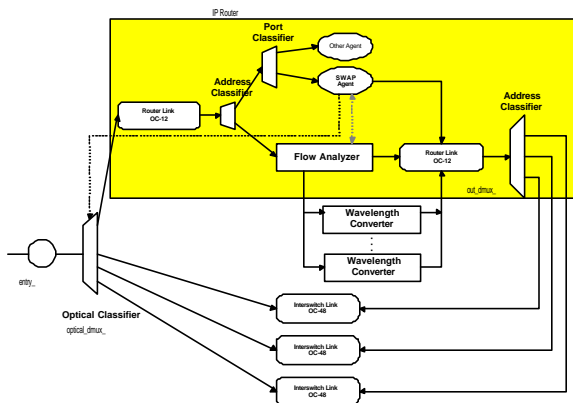


Figure 7 – NS Model of POW switch

Essential NS components of the POW switch simulation model include (Figure 8):

- Optical Classifier: models the optical switching fabric. For simulation purposes wavelengths are represented as an integer value in our specific.
- Router Links: model the bi-directional electrical data interfaces between the IP router and the fabric. The interface operates at OC-12 rates.
- Inter-switch Links: model the high speed optical links between POW switch. This interface operates at OC-48 rates.
- Wavelength Converters: simulate wavelength converter banks in the POW switches. For the simulation, a wavelength converter reassigns

the integer value in the header representing the wavelength.

- Address Classifiers: model IP router forwarders. The classifier determines which output link a packet should be directed to. Locally-destined packets are delivered to the local application.
- Flow analyzer: an X/Y classifier, where X is `HighThreshold` (10 pkts) and Y is `FlowDetectionTimer` (20 secs). The analyzer may perform flow classification based on several tuples (see below).
- SWAP agent: models signaling between the POW switches.

The POW flow analyzer is equipped to classify several traffic flows based on the following types of tuples:

- Fine Grain: classified based on the tuple: {ingress node, src ip, src tcp port} → {egress node, dst ip, dst tcp port}
- Medium Grain: {ingress node, src ip} → {egress node, dst ip}
- Coarse Grain: {ingress node} → {egress node}.
- Merged Fine Grain: * → {egress node, dst ip, dst tcp port}.
- Merged Medium Grain: * → {egress node, dst ip}
- Merged Coarse Grain: * → {egress node}

POW nodes are interconnected using NS Tcl script which encodes the vBNS backbone topology, shown in Figure 8. The vBNS network is a good example of the type of environment where POW would be useful. vBNS currently provides IP service on top of an ATM network. vBNS nodes are connected by a complete mesh of PVCs to each other. POW can replace this mesh with a more effective optical Internet.

Although the main goal of POW is to evaluate the number of wavelengths required per link, we also evaluated the overhead of SWAP protocol. LBL-PKT5 [ITA] was chosen as the traffic model.

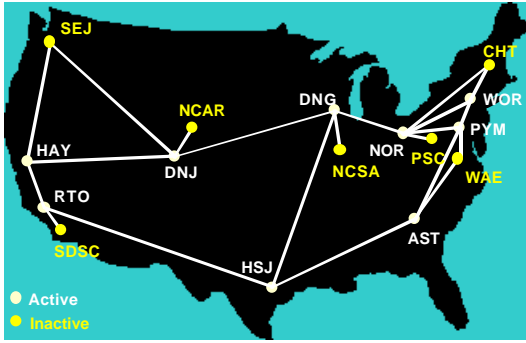


Figure 8 - vBNS Backbone Topology

To instrument the performance measure of SWAP in term of the overhead, the ratio of SWAP messages is compared to the offered traffic. Five schemes are compared: non-merged fine, medium and coarse grain flows and merged fine and coarse grain flows with different numbers of available wavelengths: 0, 4, 8, 16, 32, and 64. Default wavelength λ_0 is assumed and is not counted.

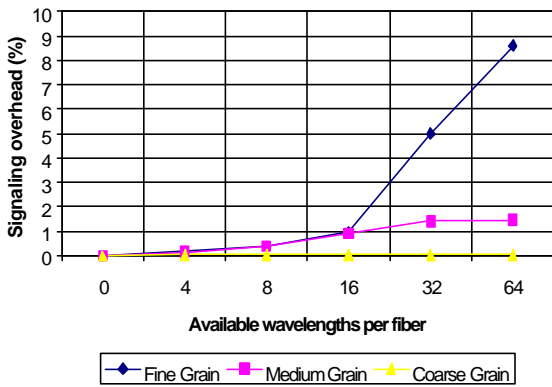


Figure 9 - SWAP overhead for non-merging flows

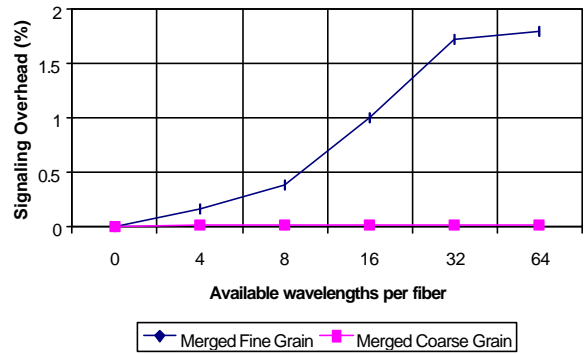


Figure 10 - SWAP overhead for merged flows

Figure 9 and Figure 10 indicate that for most cases, SWAP overhead is proportional to the number of switchable flows. For non-merged fine grain case, there is an unexpected increase in the overhead for 32 and 64 wavelengths. Figure 11 indicates that there is no significant increase in the number of flows being switched. Simulation traces show that there were a large amount of unsuccessful signaling due to the high number of switchable branches. Primarily there were not enough common wavelengths to construct the lightpaths, and SWAP incurred a high percentage of unsuccessful signaling attempts. If the flows persist after such a failure, this will significantly increase the number of SWAP messages exchanged. Similar behavior is observed for merged-flows.

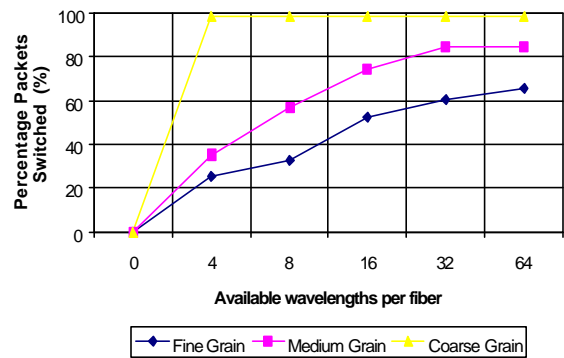


Figure 11 - Percentage of Packet Switched for non-merging flows

Figure 10 exhibits this behavior more clearly for merged-flows, where SWAP has three different impediments: insufficient wavelengths overall, sufficient wavelengths but insufficient common

wavelength along the path, and insufficient switchable flows. Both Figures 9 and 10 show three phases in the graph. For the merging case, the first phase is between 0 and 8 available wavelengths. This phase shows the overhead of SWAP when there were not enough wavelengths: the number of signaling messages grows slowly because most of the messages are SETUP and SETUP_FAIL. In this phase, most SETUP attempts fail quickly because the free wavelength sets for the links between the last hop to the previous hop were empty.

The second phase between 8 and 32 available wavelengths corresponds to the case when there were enough wavelengths (more opportunity to switch the flows), and SWAP became more active performing its signaling. However, because there were insufficient common wavelengths along the entire path, SWAP kept trying unsuccessfully. In this case, the number of SWAP messages grows more rapidly in this case, and this could become a scalability issue if SWAP were used when the ratio between Common Wavelengths available per the number of Switched Flows (CW/SF) is low. In this case the competition for the wavelengths is not self-limiting, as in the case with too few switched flows. It may be necessary to consider preemption, where new large flows cause current small switched flows to be torn down, and wavelengths reassigned. Another alternative would be to consider limiting the propagation of setup messages, relative to the number of current flows, number of wavelengths, and likely length of the path. Either of these alternatives would require augmentation of the SWAP protocol.

Last phase is between 32 and 64 available wavelengths, and corresponds to sufficient common free wavelengths, where the number of SWAP messages during signaling was bounded. In this case, most of the SETUP attempts succeed. These three phases give an insight on the scalability of SWAP in term of the messaging overhead when CW/SF is very small (approaching zero), small, and large.

CONCLUSION

The SWAP protocol is an approach to assigning wavelengths dynamically based on traffic demand, and was created to facilitate the Packet over Wavelength (POW) architecture. SWAP's wavelength assignment is similar to IP switching; it is data-driven, rather than control-driven. A notable difference is that POW introduces flow merging to reduce the number of wavelengths required. This

requires SWAP nodes to perform both active and passive roles based on their positions in the flow tree. These roles must be performed at the point where electrical signals are still available: the first hop or last hop of the lightpaths.

During POW simulation, SWAP overhead was measured under three conditions: where the Common Wavelength Per Switchable Flows (CW/SF) ratio is zero, small, and large. SWAP scales well for zero and large CW/SF. For a small CW/SF ratio, SWAP scales poorly due to the breadth of wasted signaling. This may suggest that SWAP need to be augmented to support lightpath preemption, where new larger flows cause smaller existing flows to be torn down, and wavelengths reassigned. Preemption will enable SWAP to switch the N most significant switchable flows (for N free wavelengths), where the SWAP overhead would thus be bounded.

Future study is required to measure SWAP performance under higher traffic loads. In addition, further investigation is required to determine how SWAP parameters affect its steady-state and transient performance.

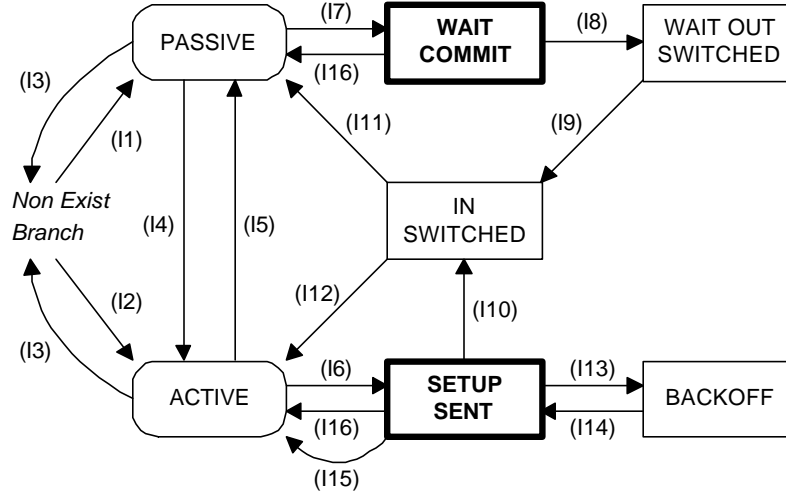
REFERENCES

- [Cal+97] R. Callon, et.al., "A Framework for Multiprotocol Label Switching", (work in progress).
- [DaDR98] B. Davie, P. Doolan, and Y. Rekhter, *Switching in IP Networks: IP Switching, Tag Switching and Related Technologies*, Morgan Kaufmann, San Francisco CA, 1998.
- [Kesh97] S. Keshav, *An Engineering Approach to Computer Networking: ATM Networks, the Internet and the Telephone Network*, Addison-Wesley, Reading MA, 1997.
- [ITA] <http://ita.ee.lbl.gov/html/traces.html>
- [LeLi93] K.C. Lee and V. O. K. Li, "A Wavelength-Convertible Optical network," *Journal of Lightwave Technology*, Vol. 11, No. 5/6, May 1993, pp. 962-970.
- [LiMc97] S. Lin and N. McKeown, "A Simulation Study of IP Switching," *Proceedings of ACM SIGCOMM 1997*, pp 15-24.

- [Mukh97] B. Mukherjee, Optical Communication Networks, McGraw-Hill, New York NY, 1997.
- [NeML98] P. Newman, G. Minshall, and T. Lyon, "IP Switching - ATM Under IP," IEEE/ACM Transactions on Networking, 6(2), April 1998, pp. 117-129.
- [NLANR] <http://www.nlanr.net/NA/Learn/aggregation.html>
- [NS] <http://netweb.usc.edu/vint>
- [Per192] R. Perlman, Interconnections: Bridges and Routers, Addison-Wesley, Reading MA, 1992.
- [RaSi98] R. Ramaswami and K.N. Sivarajan, Optical Networks: A Practical Perspective, Morgan Kaufmann, San Francisco CA, 1998.
- [Yoo96] S.J.B. Yoo, "Wavelength Conversion Technologies for WDM Network Applications," IEEE Journal of Lightwave Technology, Vol. 14 No. 6, June 1996, pp. 955-965.
- [ZDE+93] L. Zhang, S. Deering, D. Estrin, et.al. "RSVP: A New Resource ReSerVation Protocol," IEEE Network, Vol. 5, September 1993, pp. 8-18.

APPENDIX – DETAILED SWAP TRANSITION TRIGGERS AND ACTIONS

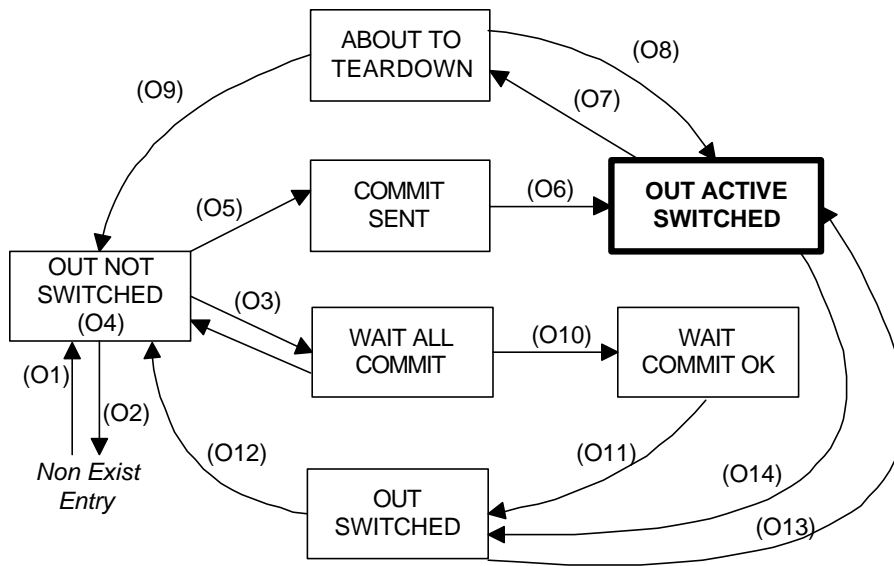
Incoming State Diagram



Triggers		Actions
I1	A packet is received and there is no branch entry associated with the packet and there is a downstream neighbor.	A new branch is created.
I2	A packet is received and there is no branch entry associated with the packet, and there is no downstream neighbor or outgoing state is either OUT_ACTIVE_SWITCHED or OUT_SWITCHED.	A new branch is created.
I3	FlowDetectionTimer expires and PacketCount is equal to zero.	Delete branch.
I4	Transition O6 or O11 happens.	The branch becomes ACTIVE and start monitoring.
I5	Transition O9 or O12 happens.	The branch becomes PASSIVE and stop monitoring
I6	FlowDetectionTimer expires and PacketCount is greater or equal to HighThreshold.	Set PacketCount to zero and send SETUP(F, \mathbf{I}_{branch}) to the upstream neighbor of the branch.
I7	Transition O3 happens and the branch is an switchable branch.	Determine $\mathbf{I}_{out} = \mathbf{I}_{in} \cap \mathbf{I}_{branch}$. If \mathbf{I}_{out} is \emptyset , reserve a wavelength converter and send SETUP(F, \mathbf{I}_{branch}) to the upstream neighbor of the branch. Else send SETUP(F, \mathbf{I}_{out}) to upstream neighbor of the branch.
I8	Receive COMMIT(F, λ_x) from the upstream neighbor.	Set $\lambda_{in} = \lambda_x$

I9	Transition O6 happens.	Setup the optical switch to redirect input from the link of the branch with λ_{in} to output link of the flow using λ_{out} . Send COMMIT_OK(F) to the upstream neighbor.
I10	Receive COMMIT(F, λ_x) from the upstream neighbor.	Set $\lambda_{in} = \lambda_x$ Setup the optical switch to redirect input from the link of the branch with λ_{in} to a wavelength converter that convert the input to λ_0 ¹¹ . Send COMMIT_OK(F) to the upstream neighbor.
I11	Receive TEARDOWN(F) and there is a downstream neighbor and this is the only branch in IN_SWITCHED state.	
I12	Receive TEARDOWN(F) and there is no downstream neighbor, or Receive TEARDOWN(F) and this is not the only branch in IN_SWITCHED state.	
I13	Receive SETUP_CONFLICT(F) or first hop cannot get the lock for the wavelengths set and BackoffCount is less than BackoffLimit.	Increase BackoffCount. Schedule BackoffTimer.
I14	BackoffTimer expires.	Send SETUP(F, λ_{branch}) to the upstream neighbor of the branch.
I15	Receive SETUP_CONFLICT(F) and BackoffCount is greater or equal to BackoffLimit.	
I16	Receive SETUP_FAIL(F)	

Outgoing State Diagram



¹¹ λ_0 is the default hop by hop wavelength, which is the wavelength used for forwarding path and signaling path.

Triggers		Actions
O1	New branch is created and there is no flow entry associated with the packet.	Creates the flow entry.
O2	Transitions I3 happens and the deleted branch is the last branch of the flow entry.	Delete the flow entry.
O3	Receives SETUP(F, \mathbf{I}_{in}) from downstream neighbor and there is at least one switchable branch and full path construction is possible ¹² .	<ol style="list-style-type: none"> Pick $\lambda_{outgoing} \in \mathbf{I}_{in}$. If the lightpath is potentially contiguous, $\lambda_{outgoing}$ will be determined later after get COMMIT(F) from the upstream. In the case of non-contiguous, pick one arbitrarily. For each switchable branch where full path construction is possible, do transition I7.
O4	Receives SETUP(F, \mathbf{I}_{in}) from downstream neighbor and there is at least one switchable branch and full path construction is not possible and PartialPathAllowed is not set.	Send SETUP_FAIL(F) to the downstream neighbor.
O5	Receives SETUP(F, \mathbf{I}_{in}) from downstream neighbor and there is no switchable branch, or there is a switchable branch, full path construction is not possible and PartialPathAllowed is true.	<ol style="list-style-type: none"> Pick $\lambda_{outgoing} \in \mathbf{I}_{in}$ arbitrarily. Send COMMIT(F, $\lambda_{outgoing}$) to the downstream neighbor.
O6	Receives COMMIT_OK(F) from downstream neighbor.	Redirect λ_0 to $\lambda_{outgoing}$.
O7	FlowActiveTimer expires and aggregate packet count is less than LowThreshold.	Schedule FlowActiveTimer.
O8	FlowActiveTimer expires and aggregate packet count is higher than LowThreshold	
O9	FlowActiveTimer expires and aggregate packet count is less than LowThreshold	<ol style="list-style-type: none"> Undo λ_0 to $\lambda_{outgoing}$ conversion. Send TEARDOWN(F) to downstream neighbor.
O10	I8 happens and the branch is the only branch in WAIT_COMMIT state.	Send COMMIT(F, $\lambda_{outgoing}$) to downstream neighbor.
O12	I11 happens and aggregate packet count is less than LowThreshold.	<ol style="list-style-type: none"> Send TEARDOWN(F) to downstream neighbor. Undo $\lambda_{incoming}$ to $\lambda_{outgoing}$ redirection.
O13	I11 happens and aggregate packet count is higher or equal to HighThreshold.	<ol style="list-style-type: none"> Undo $\lambda_{incoming}$ to $\lambda_{outgoing}$ redirection. Convert λ_0 to $\lambda_{outgoing}$.
O14	I10 happens	Cancel FlowActiveTimer

¹² That is $\mathbf{I}_{in} \cap \mathbf{I}_{branch} \neq \emptyset$ or there is a free wavelength converter and WavelengthConvertEnable is true.