Simple Wavelength Assignment Protocol[†‡]

Stephen Suryaputra[*a], Joseph D. Touch[b], and Joseph Bannister[b]

[a]Nortel Networks

[b]USC/Information Sciences Institute

ABSTRACT

IP routers can be coupled with wavelength-selective optical cross-connects to support 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 pass directly through the optical cross-connect, resulting in very high throughput and low delay routing. 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 conversions are expensive, wavelength "label" swapping is not easily done. Wavelength "label" assignments must consider these limitations to be practical in an optical environment. The performance of an instance of this approach, called Packet over Wavelengths (POW) has been simulated and studied. A new signaling protocol, Simple Wavelength Assignment Protocol (SWAP) is devised to be POW signaling protocol. SWAP takes into account the optical device limitations, and is designed to minimize wavelength conversions, utilize wavelengths with the merging of flows, and reduce the reconfiguration of optical switches. SWAP, to our knowledge, is the first approach to combine signaling and wavelength assignment in an on-line protocol. This paper describes high level SWAP design challenges, decision, and overhead.

Keywords: Optical Networks, Wide-Area Lightwave Networks, Wavelength Division Multiplexing (WDM), Protocol, Routing and Wavelength Assignment (RWA), Optical Network Control, IP Switching, MPLS.

1. INTRODUCTION

Internet protocol (IP) routers can be coupled with wavelength-selective optical cross-connects to enable 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 routing can be achieved when packets bypass the IP forwarding process by passing directly through the optical cross-connect. Although 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 (ATM VPI:VCI space is 2^{28} and MPLS label space is 2^{16}). 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 (in the order of 80,000 flows [14]). Wavelength converters (an optical corollary to label swapping) are expensive [21]. 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 cross-talk and loss [17]. 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 [2]. This study is based on a signaling protocol created to dynamically configure the lightpath for flows. This protocol is called the Simple Wavelength Assignment

[†] This research is partially supported by the Defense Advanced Research Projects Agency (DARPA) through contract MDA972-99-C-0022 of the Maryland Procurement Office. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Maryland Procurement Office, DARPA, or the U.S. Government. The authors can be contacted at USC/ISI, 4676 Admiralty Way, Marina del Rey, CA, 90292-6601, U.S.A., or by electronic mail at touch@isi.edu, joseph@isi.edu, or ssuryapu@nortelnetworks.com.

[‡] An earlier version of this paper appeared as a USC/ISI Research Report [19].

[*] Work performed while a graduate student at USC/ISI.

Protocol (SWAP). SWAP modeling and simulation is performed on the VINT/ns simulator [NS]. POW has two forwarding paths: regular (slow) and cut-through (fast) path. Initially all flows are forwarded on slow path while SWAP tries to identify a flow. When a flow is detected, SWAP configures the fast-path and forwards the packets belonging to that flow on the configured path. During signaling, all the packets of the flow are still forwarded on slow-path. A dedicated wavelength is used as the slow path and the signaling path. Although the essential idea of POW is not new (it is basically IP switching [13]), POW takes further steps beyond IP switching to increase the applicability of the approach to optical backbone. In addition, we believe that SWAP is the first approach to combine signaling and wavelength assignment in an on-line protocol.

2. 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 hop count (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 implicitly in the physical layer instead of in the link or network layer by the wavelength of the packet signal. Another difference between POW and label switching approaches is that label switching never has a notion of label unification that is essential to use wavelengths as labels. 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 [5]. Among these, the most notable are MPLS and IP Switching. MPLS [4] is the IETF standard for label switching and its principal operation is similar to Tag Switching; this discussion thus regards these two approaches as the same. The primary difference between MPLS and IP Switching is the process of assigning labels to IP traffic. MPLS assigns flows based on routing protocol events; IP switching is based on the actual traffic seen on the network. The taxonomy described in [5] thus considers MPLS 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 over abundance of switchable flows and signaling latencies, POW takes further steps by coarsely aggregating flows to and merging flows. By aggregating flows, POW reduces the number of flows handled by each switch and minimizes the impact of signaling latency. By merging flows, POW localizes signaling dynamics to the nodes near the edges of the network.

There have been several recent proposals optical switching techniques, both in the academic as well as industry (standard bodies). Optical Burst Switching (OBS) [16] and Terabit Burst Switching [20] took fast circuit switching approach in early 1980's. Burst switching uses pre-reservation of optical paths by sending a control message just before the burst of packets. Burst switching maintains statistical multiplexing property of packet switching while keeping the burst in the optical domain. A practical implication of applying burst switching to the current Internet is that either the boundary switch (the switch sits between electrical and optical domains) needs to have adequate buffering to detect and collect the burst, or the session source, knowing how big the burst will be, can function as the originator of the control message. Another problem of burst switching that precludes its use for POW is the probability of lightpath blocking, as it implies buffering in intermediate switches, changing the wavelength channel, deflection routing, or dropping. Our current study assumes that optical switches are slow, wavelength conversions are expensive, and the burst size of domain-to-domain traffic is large. Because optical switches are slow, then the delay between control message and the burst might need to be large. This requires even bigger buffers at the boundary between electrical and optical. Moreover, if wavelength conversion is expensive and the burst size is large, it might be better to construct the longest possible contiguous lightpath (to minimize wavelength conversions) and to forward the burst conventionally while waiting for signaling to complete. SWAP is designed to match more closely with POW requirements and assumptions.

Another approach is sub-carrier multiplexing (SCM) [3]. SCM operation is similar to MPLS, inserting a small header before the IP packet. SCM node examines the header by doing optical to electronic conversion. Once the header has been recognized, a rapid tunable laser is configured based on the header lookup information, the old header is stripped and a new header is inserted. SCM requires (and assumes) that the optical to electronic to optical (O-E-O) conversion happens only for the small header (or tag). SCM thus avoids buffering of the whole packet eliminating the need for large optical buffers or O-E-O conversion of the payload. The key difference between SCM and POW are: 1) POW doesn't require a rapidly tunable laser, and 2) POW introduces the notion of merging traffic flow optically. However, SWAP approach of assigning wavelengths can be applied for SCM.

Several IETF working groups have been exploring work on IP over optics, such as MPLS, TEWG, and IPO BOF. There are a number of Internet drafts covering various topics such as a framework for IP and optical interworking, extensions to OSPF and IS-IS, and extensions to RSVP [22] and CR-LDP [7]. Most of the IETF proposals use MPLS-based protocols to distribute optical link characteristics (wavelengths) and to control optical switches. They are extensions to MPLS Traffic Engineering proposals. Other forums are OIF and ODSI, which have similar proposal for signaling. POW focuses on creating effective optical shortcuts based on the on-line traffic demand, thus it is closer to IP switching. SWAP is the traffic monitor and the label distribution protocol for POW. Even though SWAP operation is similar to MPLS Label Distribution Protocol (LDP), it introduces the new concept of label unification.

3. HIGH LEVEL DESIGN

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

- ***Control Processor***: an IP router equipped with high-speed 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 (with increased complexity).
- ***Optical Fabric*** (λ switch): 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 Mergers***: merges optical signal so that two optical signals on the same wavelength that come from different input ports but destined to the same output port can be merged on one wavelength channel. Merging device can be viewed as path contention resolution strategy of POW switch and it enables POW to support merging of traffic flows in optical domain.
- ***Wavelength mux/dmux***: redirects a specific wavelength to a particular fabric, and merges the output of the fabrics to the external fibers. These are special cases of statically-configured optical switching elements.

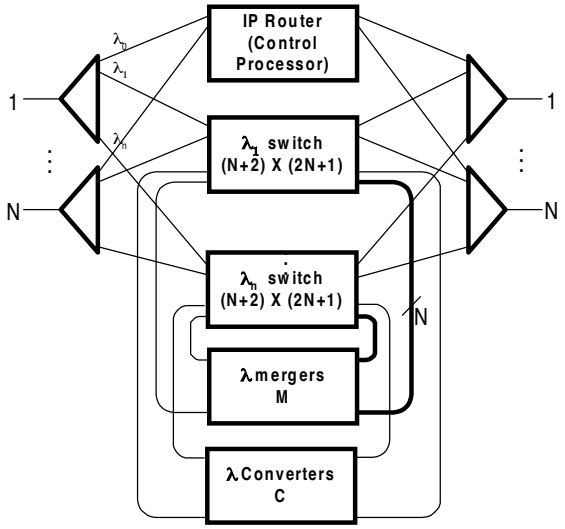


Figure 1 - POW switch architecture

Figure 1 shows an N×N POW switch where there are n optical switches, M wavelength mergers and C wavelength converters. Each optical switch is an (N+2)×(2N+1) expander and is responsible of a specific wavelength interconnect. (N+2) input connects N demuxes, 1 wavelength mergers and 1 wavelength converters, and (2N+1) output connects N outputs to the muxes, N possible contending inputs to the mergers and 1 need-to-be-converted inputs to the converters. The arrangement of the POW components follows *share-with-local wavelength-convertible* switch architecture [9] to reduce cost. Therefore it is possible that M and C are less than N. It is important to note that POW requires one hop-by-hop default wavelength λ_0 for signaling and regular forwarding. This wavelength is connected to the control processor. The details of the POW components, as well as their configuration, are under further study and beyond the scope of this paper.

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 protocol.

For 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), a reliable transport above AAL5 [8]. The telephone Signaling System 7 (SS7) [11] and LDP-based MPLS signaling [1][7] make the same choice. SWAP establishes per-neighbor reliable transport connections (like TCP or SCTP [18] connections), over which the signals are sent. These connections enable SWAP to determine whether it is running on 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). Neighbor discoveries as well as the underlying reliable transport are not the main aspect of SWAP therefore it is not part of the design space.

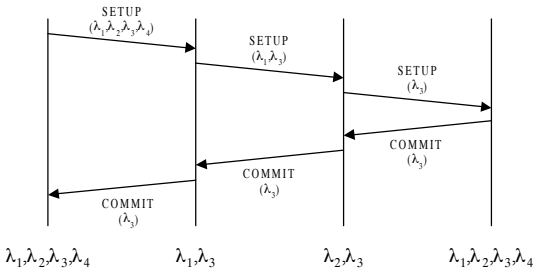


Figure 2 - Time sequence diagram of first-hop-initiated signaling

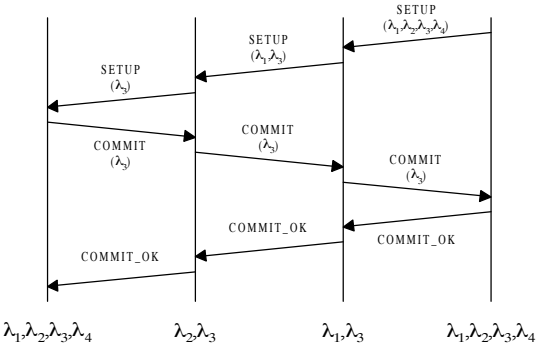


Figure 3 – Time sequence diagram of last-hop-initiated signaling

Constructing the contiguous light path as far as possible requires SWAP to pick a common free wavelength along the flow path, therefore SWAP 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 distinguishes SWAP from label swapping, 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 a wavelength. The wavelength resource must be also locked during signaling to avoid race-condition. SWAP could potentially do *crank-back mechanism* to eliminate locking, i.e. the free wavelength can be taken by another flow when it is time to setup (COMMIT) the switches. The rationale behind not doing crank-back is: 1) it increases expected signaling delay (due to back-and-forth signaling) therefore the flow might not be able to fully utilize the path; 2) it might not necessary since the target flow aggregation of POW is very coarse (between electrical domains) and the number of wavelengths per link is increasing significantly.

Next SWAP decides where to initiate the signaling. 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 2.

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 flow merging. 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 setup. The first sequence is similar to the first-hop initiated, 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 emulate 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 2.

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. There are common available wavelengths for all examples below. Specific case when wavelength conversion is needed will be described later in implementation section.

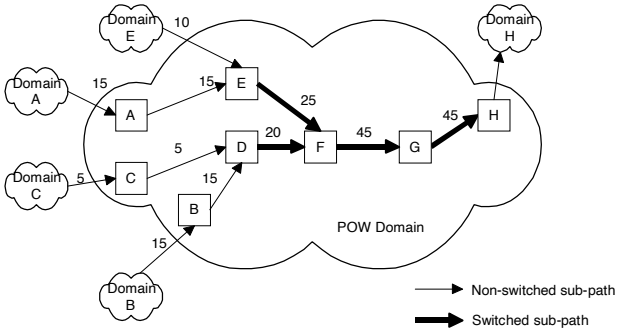


Figure 4 – Flow aggregation scheme

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

Table 1 – Free wavelengths at Figure 4 switches

Suppose SWAP was set to use X/Y classifier [10] and 20 packet per second (pps) was 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 λ_{GH} and send SETUP(F_1, λ_{GH}) to G. Upon receiving that SETUP, G does the wavelength set intersection $\lambda_x = \lambda_{GH} \cap \lambda_{FG} = \{\lambda_2,\lambda_3,\lambda_4\}$, lock λ_{FG} and sends SETUP(F_1, λ_x) to F. F does the same intersection $\lambda_y = \lambda_x \cap \lambda_{DF} = \{\lambda_2\}$ and $\lambda_z = \lambda_x \cap \lambda_{EF} = \{\lambda_3,\lambda_4\}$. Then it will send SETUP(F_1, λ_y) to D and SETUP(F_1, λ_z) to E. Both D and E know that they are the first hops for that path because there is no upstream neighbor for the path, or there is no incoming branches that have a high incoming throughput. Knowing that they are the first path, D will send COMMIT(F_1, λ_2) and E will send COMMIT(F_1, λ_3). Meanwhile, F waits for responses from both. Upon receiving the responses, F picks λ_3 because it is an element of λ_{EF} and E contributed more to the aggregate throughput than D. F removes λ_3 from the set λ_{EF} and λ_2 from the set λ_{DF}, and sends COMMIT(F_1, λ_3) to G. G removes λ_3 from the set λ_{FG}, unlocks it and forwards COMMIT(F_1, λ_3) to H. Then, H removes λ_3 from the set λ_{GH}, unlocks it, configuring its optical switch and flow converter to convert λ_3 back to the default wavelength λ_0, and send COMMIT_OK(F_1) to G.

Upon receiving COMMIT_OK(F_1) from H, G sends COMMIT_OK(F_1) to F, and F extends the message to both D and E. When E receives COMMIT_OK(F_1) from F, it starts sending the flow through a wavelength converter that converts default wavelength λ_0 to λ_3. D does the same conversion for λ_0 to λ_2. Wavelength from branch E, λ_3, was selected during COMMIT phase because E contributes more to the aggregate throughput and it is more likely to stay that way. 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 SETUP(F_1, $\boldsymbol{\lambda}_{BD}$) to B. Current design of SWAP doesn't attempt to maintain wavelength continuity after the lightpath has been set up, however if if λ_2 is an element of $\boldsymbol{\lambda}_{BD}$ then D will send SETUP(F, {λ_2}). B determines that it is the first hop and sends COMMIT(F_1, λ_3). Upon receiving from B, D configures its optical switch and sends 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 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 TEARDOWN(F_1), TEARDOWN(F_2), …, TEARDOWN(F_n) where $F_{1\text{-}n}$ are the switched flows coming from the neighbor. If the failed neighbor is 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 with insufficient incoming throughputs to trigger a flow. Last hop is similarly 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 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 te 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 neither upstream nor downstream neighbors at a node, SWAP will be disabled there 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).

4. IMPLEMENTATION

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

- SETUP(F, $\boldsymbol{\lambda}$). 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 recepients 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 on 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 resources (wavelengths or devices needed to setup the lightpath). 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.

Current SWAP experiment 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. Discussions in the following paragraph don't elaborate the detailed SWAP state machine since the detailed description is included in the appendix.

To understand SWAP state transitions, some definitions are provided for the further discussion. 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 combinations of lightpaths: contiguous or non-contiguous, and partial or full. A *contiguous lightpath* is a ligthpath 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 true. 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.

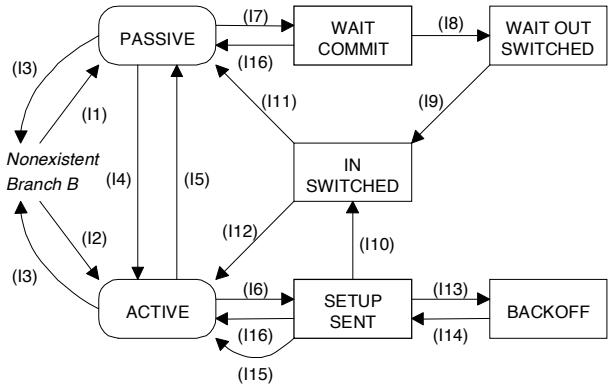


Figure 5 – Incoming state transition

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 criteria:

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 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 λ_B and send SETUP(F_1, λ_B) to its upstream neighbor, S_P. If λ_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 F1 and monitor it for another detection window.

The upstream neighbor S_P (passive role) upon receiving SETUP(F, λ_{in}) 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 λ_n, assign $\lambda = \lambda_{in} \cap \lambda_n$ and send SETUP(F, λ) to the upstream neighbor. Here the outgoing state of F becomes `WAIT_ALL_COMMIT` (Figure 6). However, if **IC3** and **IC5** are true, S_P will pick $\lambda_{out} \in \lambda_n$ 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).

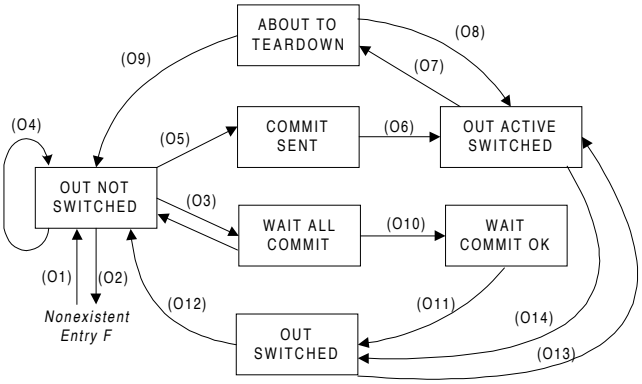


Figure 6 - Outgoing state transition

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 branch. 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. There is a possibility that each branch picks different wavelength therefore creates potentially more wavelength discontinuity in the lightpath. To avoid this problem, SWAP could be modified so that λ in COMMIT(F, λ) is a set and S_P picks λ_x such that λ_x is the common element of all the branch sets.

If the criteria **IC3**, **IC4**, and **IC5** cannot be met concurrently, S_{Pn} will send a SETUP_FAIL(F) message downstream. This message will not be propagated by 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 SWAP node S_n 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.

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 swiched 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.

5. 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 [15]. Essential NS components of the POW switch simulation model include (Figure 8):

- ***Optical Classifier***: models the optical switching fabric. For simulation purposes wavelenghts 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.

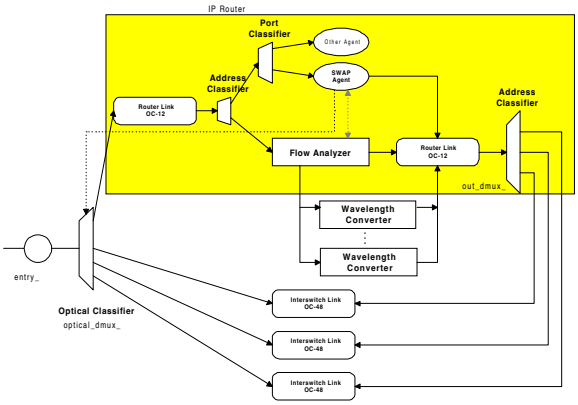


Figure 7 – NS model of POW switch

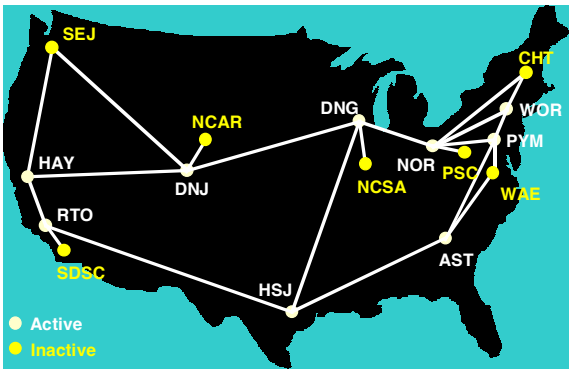


Figure 8 - vBNS backbone topology

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

- Fine Grain: {ingress switch, src ip, src tcp port}→{egress switch, dst ip, dst tcp port}
- Medium Grain: {ingress switch, src ip}→{egress switch, dst ip}
- Coarse Grain: {ingress switch}→{egress switch}.
- Merged Fine Grain: *→{egress switch, dst ip, dst tcp port}.
- Merged Medium Grain: *→{egress switch, dst ip}
- Merged Coarse Grain: *→{egress switch}

POW nodes are interconnected using NS Tcl script which encodes the vBNS backbone topology as of September 18, 1998, 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 [6] was chosen as the traffic model. In the interest of understanding SWAP overhead, we intentionally setup the simulation so that there always enough wavelength converters and mergers.

To instrument the performance measure of SWAP in term of the overhead, the number of SWAP messages is compared to the offered traffic and expressed as a ratio. Five schemes are evaluated: 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. The default wavelength λ_0 is implicit and is not counted.

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 large amounts of unsuccessful signaling due to the high number of swichable 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.

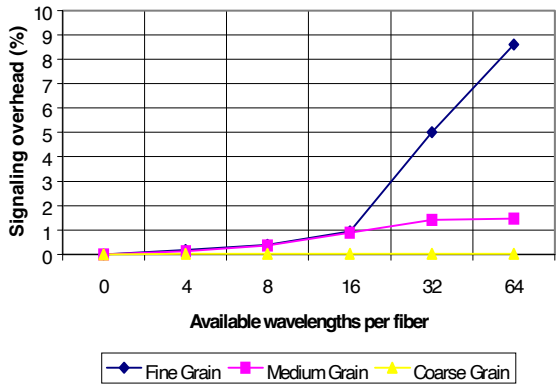


Figure 9 - SWAP overhead for non-merging flows

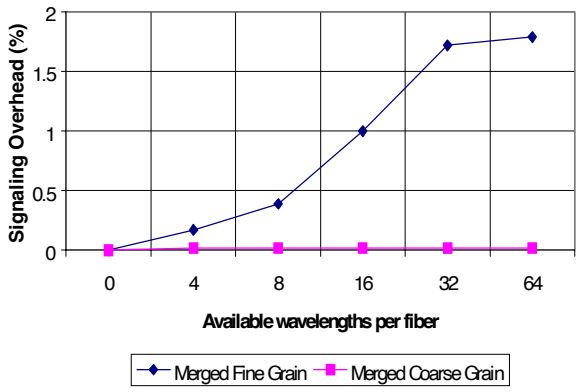


Figure 10 - SWAP overhead for merged flows

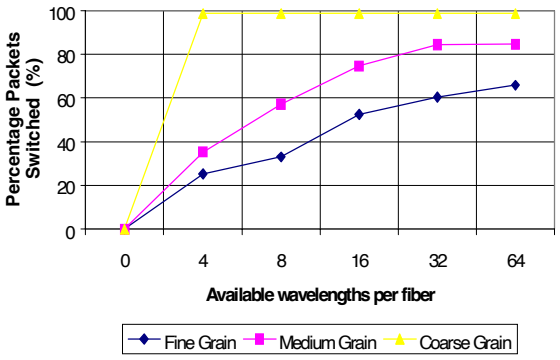


Figure 11 - Percentage of packets swithed for non-merged flows

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 enture path, SWAP kept trying unsuccessfully. In this case, the number of SWAP messages grows more rapidly in this case, and this could affect the scalability if SWAP were used when the ratio between Common Wavelengths available per the number of Switched Flows (CW/SF) is small. 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 limit 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. This last behavior also supports our rationale before that SWAP locks the resources (wavelength sets) and avoids crank-back mechanism when CW/SF is large.

6. 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 wavelength assignment is similar to IP switching; it is data-driven, rather than control-driven. A notable difference is that SWAP 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. SWAP overhead is the first important measure of its performance because of the dynamics of the traffic flows contribute to the number of signaling messages being exchanged.

Future study is required to measure SWAP overhead under higher traffic loads. In addition, further investigation is required to determine how SWAP parameters affect its steady-state and transient performance. SWAP performance under different traffic models and different flow detection (considering flow bandwidth usage) mechanism is not known at the time of this publication. These are certainly areas for further study.

ACKNOWLEDGEMENTS

The authors thank the anonymous referees for their comments. They also thank Utham Kamath for fruitful discussions.

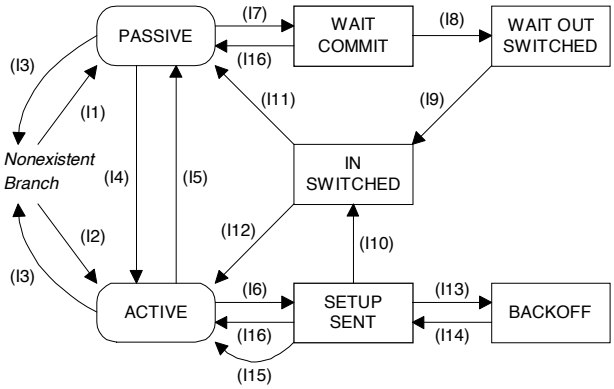
REFERENCES

1. L. Anderson, et.al, “LDP Specification,” (work in progress).
2. J. Bannister, J. Touch, A. Willner, S. Suryaputra, “How Many Wavelengths Do We Really Need in an Internet Optical Backbone?,” *Protocols for High-Speed Networks VI*, Kluwer, 1999, pp. 43-60.
3. D.J. Blumenthal, et.al., “All-Optical Label Swapping with Wavelength Conversion for WDM-IP Networks with Subcarrier Multiplexed Addressing,” *IEEE Photonics Technology Letters*, Vol. 11, No. 11, Nov. 1999, pp. 1497-1499.
4. R. Callon, et.al., “A Framework for Multiprotocol Label Switching,” (work in progress).
5. B. Davie, P. Doolan, and Y. Rekhter, *Switching in IP Networks: IP Switching, Tag Switching and Related Technologies*, Morgan Kaufmann, San Francisco CA, 1998.
6. http://ita.ee.lbl.gov/html/traces.html
7. B. Jamoussi (editor), “Constraint-Based LSP Setup using LDP,” (work in progress).
8. S. Keshav, *An Engineering Approach to Computer Networking: ATM Networks, the Internet and the Telephone Network*, Addison-Wesley, Reading MA, 1997.
9. 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.
10. S. Lin and N. McKeown, “A Simulation Study of IP Switching,” *Proceedings of ACM SIGCOMM 1997*, pp 15-24.
11. A. R. Modarressi and R. A. Skoog, “Signaling System No. 7: A Tutorial,” *IEEE Communications Magazine*, July 1990, pp. 19-35.
12. B. Mukherjee, *Optical Communication Networks*, McGraw-Hill, New York NY, 1997.
13. P. Newman, G. Minshall, and T. Lyon, “IP Switching - ATM Under IP,” *IEEE/ACM Transactions on Networking*, 6(2), April 1998, pp. 117-129.
14. http://www.nlanr.net/NA/Learn/aggregation.html
15. http://netweb.usc.edu/vint
16. C. Qiao and M. Yoo, “Optical burst switching (OBS) – a new paradigm for an Optical Internet,” *Journal of High Speed Networks (JHSN)*, vol. 8, no. 1, pp. 69-84, 1999.

17. R. Ramaswami and K.N. Sivirajan, *Optical Networks: A Practical Perspective*, Morgan Kaufmann, San Francisco CA, 1998.
18. R. R. Stewart, et.al., "Stream Control Transmission Protocol," (work in progress).
19. S. Suryaputra, J. Touch, J. Bannister, "Simple Wavelength Assignment Protocol," *ISI Research Report*, ISI/RR-99-473, Oct., 1999.
20. J.S. Turner, "Terabit burst switching," *Journal of High Speed Networks (JHSN)*, vol. 8, no. 1, pp. 3-16, 1999.
21. 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.
22. 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

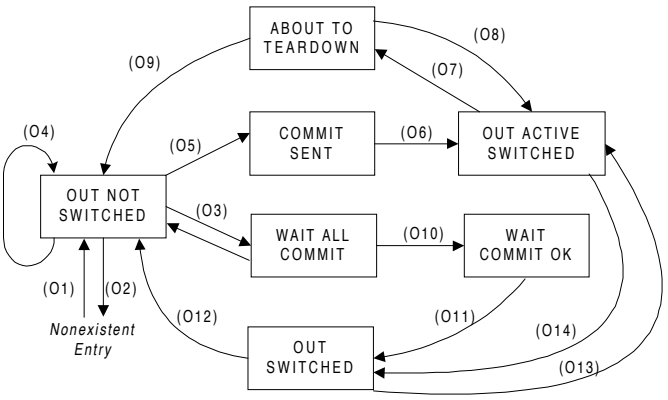
Incoming State



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, $\boldsymbol{\lambda}_{branch}$) to the upstream neighbor of the branch.
I7	Transition O3 happens and the branch is a switchable branch.	Determine $\boldsymbol{\lambda}_{out} = \boldsymbol{\lambda}_{in} \cap \boldsymbol{\lambda}_{branch}$. If $\boldsymbol{\lambda}_{out}$ is $\varnothing$, reserve a wavelength converter and send SETUP(F, $\boldsymbol{\lambda}_{branch}$) to the upstream neighbor of the branch. Else send SETUP(F, $\boldsymbol{\lambda}_{out}$) 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. 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, $\boldsymbol{\lambda}_{banch}$) 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



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, $\boldsymbol{\lambda}_{in}$) from downstream neighbor and there is at least one switchable branch and full path construction is possible (That is $\boldsymbol{\lambda}_{in} \cap \boldsymbol{\lambda}_{branch} \neq \varnothing$ or there is a free wavelength converter and `WavelengthConvertEnable` is true).	Pick $\lambda_{outgoing} \in \boldsymbol{\lambda}_{in}$. 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, $\boldsymbol{\lambda}_{in}$) 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, $\boldsymbol{\lambda}_{in}$) 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.	Pick $\lambda_{outgoing} \in \boldsymbol{\lambda}_{in}$ 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	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.
O11	Receive COMMIT_OK(F) from downstream	Redirect λ_{in} to $\lambda_{outgoing}$
O12	I11 happens and aggregate packet count is less than `LowThreshold`.	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`.	Undo $\lambda_{incoming}$ to $\lambda_{outgoing}$ redirection. Convert λ_0 to $\lambda_{outgoing}$.
O14	I10 happens	Cancel `FlowActiveTimer`