Datagram Forwarding via Stateless Internetwork Switching

Gregory G. Finn
Rodney Van Meter, Craig Milo Rogers
University of Southern California/Information Sciences Institute
4676 Admiralty Way
Marina Del Rey, CA 90292
(310) 822-1511

Abstract

Virtual-circuit switching achieves high forwarding performance via the combination of a hardware frame forwarding path and link-layer connection state that is stored in each switch along a virtual circuit. A similar result can be achieved when forwarding datagrams across an internetwork by uncoupling forwarding from hop-by-hop routing. That uncoupling is achieved by using a specialized network-layer protocol to wrap the datagram of another network-layer protocol with a source route that is derived by consulting hop-by-hop routing tables. Periodically re-deriving these source routes achieves the adaptability needed to maintain compatibility with connectionless datagram routing.

This paper proposes source routing as an alternative to distributed-state switching protocols for forwarding IP traffic.[1]

1 Introduction

It has been strongly argued that the hop-by-hop routing of datagrams is a significant performance obstacle because the data forwarding path is not purely a hardware function [1][2]. A principal motivation for adopting IP over ATM is the promise of improved performance that results when hop-by-hop datagram routing is replaced by switched forwarding across virtual circuits. However, the Internet is based upon IP, which is connectionless. Moreover, the Internet has a diverse and evolving link-layer domain.

Performance, diversity and evolution concerns would be satisfied by a datagram switching methodology that operates at the network layer. A *network-layer switch* forwards datagrams from an incoming link-layer interface to an outgoing link-layer interface, where incoming and outgoing link-layer technologies can be different. To the extent that switching replaces hop-by-hop routing, forwarding performance in the internetwork becomes increasingly dependent upon switch performance and less dependent on router performance.

A network-layer switch requires two pieces of state information to forward a datagram: the outgoing interface and the outgoing link-layer header for that datagram. IP Switching [2][3] and Tag Switching [4][5] locate a portion of that state in each cell (datagram) and another portion in each switch.

Both IP Switching and Tag Switching associate with each cell (datagram) an identifier that names the circuit or route along which a cell (datagram) is forwarded. That identifier is retrieved upon reception to index a forwarding table in the switch. Data retrieved from that table determines: (1) the choice of output port for the cell (datagram) being switched, (2) the new identifier value to be placed into the cell (datagram) for the next hop, and (3) the outgoing link-layer header.

The forwarding state necessary to set up these circuits or routes must be correctly distributed and maintained within the switches to properly forward datagrams.

1.1 Source-Routed Network-Layer Switching

The approach developed in this paper differs by eliminating the need to distribute and store forwarding state within switches. Instead, complete forwarding state is stored within each datagram as a source route. Consequently, that forwarding state need be maintained only at the head of the source route, switches do not need a forwarding state table, and are thereby simplified and freed from the scaling constraints associated with such tables. Source routing trades off these advantages against increased datagram length.

A source-routed switch is essentially stateless. It retains no routing state and so is logically independent from its associated router(s). This makes the switch inherently resistant to attacks made upon routers. Router failure, swap-out or restart need not affect switch forwarding function. Intermediate routers along a source route may all fail, but as long as links and switches function, that source route can still be used to forward datagrams through the switching fabric. This enhances internetwork survivability.

1. This research was sponsored by the Defense Advanced Research Projects Agency under Contract No. DABT63-93-C-0062. Views and conclusions in this paper are the authors' alone.

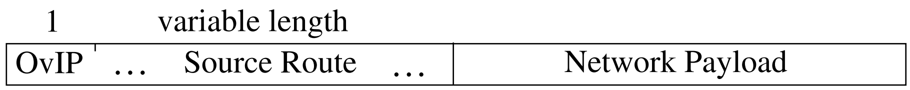

Figure 1. OvIP Datagram

2 Outline of OvIP and Switch Function

OvIP (Overpass IP) is a source-routed network-layer protocol designed to enable the forwarding of datagrams via switching. A source route is associated with a network-layer datagram by preceding that datagram with a header that is defined for that purpose. The presence of an OvIP header differentiates datagrams that are to be forwarded by switching, hereafter called OvIP datagrams, from those that require hop-by-hop routing.[2] See Figure 1.

OvIP source routes are gathered by route-tracing datagrams that interrogate OvIP router/switches. The state information retrieved includes both outgoing link and link-layer encapsulation data. This information is privately defined by each router/switch, reflecting its particular architecture and topology. It may also include quality-of-service or other flow-specific data to be used by the switch when forwarding a datagram.

2.1 Preliminaries

We assume that within end-to-end Internet routes $\boldsymbol{src}, \boldsymbol{r_1} \ldots \boldsymbol{r_n}, \boldsymbol{dst}$ there exist stable sub-routes $\boldsymbol{r_i} \ldots \boldsymbol{r_k}$ (stability of routes is discussed in Section 5). A router or host $\boldsymbol{r_i}$ can learn the forwarding and link-layer encapsulation decisions that would be taken by routers $\boldsymbol{r_{i+1}} \ldots \boldsymbol{r_{k-1}}$ when they forward datagrams to $\boldsymbol{dst}$. That information constitutes a source route $S(\boldsymbol{r_i}, \boldsymbol{r_k})$. If a datagram sent from $\boldsymbol{r_i}$ is accompanied by $S(\boldsymbol{r_i}, \boldsymbol{r_k})$, intermediate routers $\boldsymbol{r_{i+1}} \ldots \boldsymbol{r_{k-1}}$ need not reexamine that datagram to route it. A network-layer switch could automatically forward that datagram from incoming to outgoing link. If a stable path extends from end-to-end, datagrams could be switched from $\boldsymbol{src}$ to $\boldsymbol{dst}$.

To retain Internet compatibility, forwarding must adapt to the dynamics of network-layer routing. To achieve this the host or router at the head of an OvIP source route, called an *initiator*, caches the source route, treating it as *soft state* [7]. A source route $S(\boldsymbol{r_i}, \boldsymbol{r_k})$ is maintained by its initiator, $\boldsymbol{r_i}$, which periodically re-derives and updates it. An OvIP source route ends at a host or router $\boldsymbol{r_k}$ called the *terminator*. See Figure 2.

Assume that a datagram sent by $\boldsymbol{src}$ to $\boldsymbol{dst}$ follows the path $\boldsymbol{r_1}$ through $\boldsymbol{r_n}$ and that along this path is a sequence of router/switches $\boldsymbol{r_i} \ldots \boldsymbol{r_{k-1}}$ that support OvIP. An OvIP source route can then extend across $\boldsymbol{r_i} \ldots \boldsymbol{r_k}$. Assume that $\boldsymbol{r_i}$ discovers the source route $S(\boldsymbol{r_i}, \boldsymbol{r_k})$ toward $\boldsymbol{dst}$. $S(\boldsymbol{r_i}, \boldsymbol{r_k})$ may then be associated by $\boldsymbol{r_i}$ with a routing table entry for $\boldsymbol{dst}$. Subsequently, when $\boldsymbol{r_i}$ routes a datagram to $\boldsymbol{dst}$, it can wrap that datagram with an OvIP header containing $S(\boldsymbol{r_i}, \boldsymbol{r_k})$ and forward it. Eventually, the $\boldsymbol{r_{k-1}}$ will receive that OvIP datagram, whereupon as terminator it will forward the original unwrapped datagram to $\boldsymbol{r_k}$.

Two details concerning Figure 2 bear mentioning. First, an OvIP source route may extend wherever OvIP support is present in a contiguous set of network-layer switches and their associated routers. Thus it can extend across AS (Autonomous System) boundaries. Second, although an OvIP source route can be particular to a transport-layer flow, OvIP is primarily a network-layer mechanism and should not be confused with the function of a flow label in IPv6 [7][8].

2.2 OvIP Router/Switch Function

Two classes of datagrams arrive at an interface to an OvIP switch: (1) OvIP datagrams, and (2) datagrams that require router service.

2.2.1 OvIP Datagrams

The first payload octet of an OvIP datagram is identified by the OvIP protocol value. Following that is a sequence of forwarding directives that comprise a variable-length source route. Each OvIP router/switch privately defines a sequence of octets representing its particular forwarding state. That state includes the outgoing interface, necessary outgoing encapsulation

2. It is possible to associate source-routed switching with the LLC, rather than the network-layer, by extending the SNAP definition [6]. That is inappropriate for some networks.

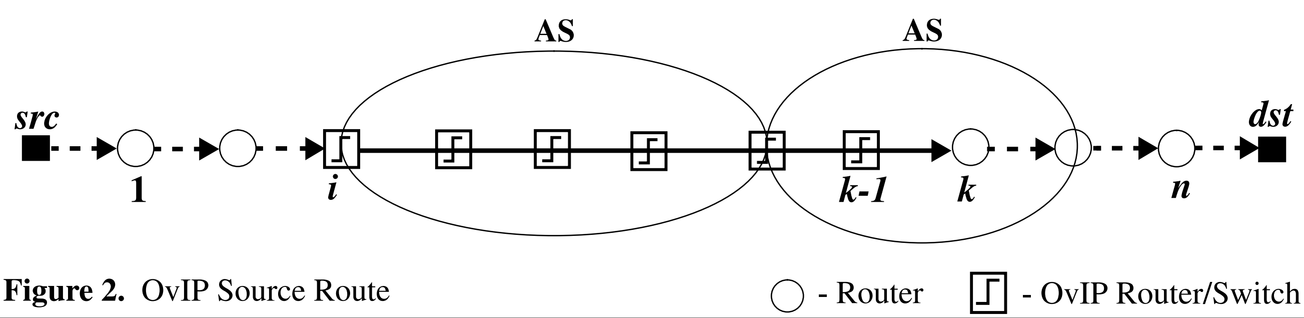

Figure 2. OvIP Source Route

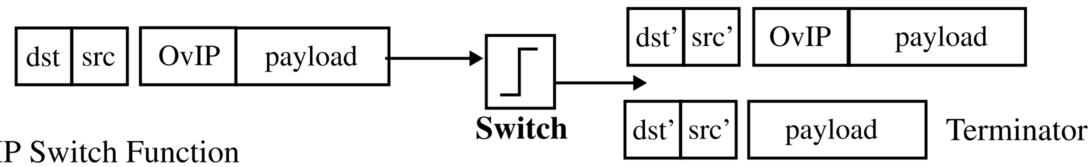

Figure 3. OvIP Switch Function

state and indication whether or not this is the terminating OvIP hop.

The forwarding state used by an OvIP switch begins in the second octet of the datagram. Subsequent to a datagram's being switched, but before it is transmitted, the forwarding state used by this switch is removed from the head of the source route. If this is the terminating hop, the leading OvIP protocol octet is also removed. Thus, after termination all OvIP data is removed, restoring the datagram to its original format. See Figure 3.

2.2.2 Non-OvIP Datagrams

Non-OvIP datagrams that arrive at the interface to an OvIP switch must be either processed locally at the interface or by a router. One approach to associate a router with a switch is to devote one switch port to the router. See Figure 4. Non-OvIP datagrams may then be treated as if they were OvIP datagrams by prefixing the portion of the incoming frame to be forwarded with a terminating OvIP source route directing the datagram to the router port. In this configuration the router can transmit across any switch interface either a non-OvIP or OvIP datagram by controlling whether or not it puts an OvIP header on its outgoing datagrams.

3 Deriving OvIP Source Routes

OvIP source routes may be derived for differing network layer protocols or flows. The particular method outlined here derives OvIP source routes from the hop-by-hop routing infrastructure supplied for IPv4 or IPv6. When an OvIP source route for a destination is derived, the source route traces the route recently used to reach that destination.

The contents of a source route S(r_i, r_k) may be obtained via a mechanism analogous to the *traceroute* utility. A new ICMP *Route-Ping* message can be defined for IPv4 [9][10] or IPv6 [11] that contains (1) the header of a datagram for which a source route is sought, (2) the source route gathered so far, (3) a hop count, and (4) the maximum transmission unit (MTU) supported so far. Route-Ping would only be implemented by OvIP router/switches.

An initiator determines whether there exists an OvIP source route toward ***dst*** and the MTU it supports by issuing a Route-Ping. Router r_i consults its routing table for ***dst***, to determine the routing directive needed to reach the next-hop switch at r_{i+1}, places the source route S(r_i, r_{i+1}) inside the Route-Ping datagram, increments the source route hop count, inserts the MTU supported across S(r_i, r_{i+1}), and forwards the Route-Ping datagram to r_{i+1}. This procedure iterates, extending the source route as the Route-Ping is forwarded hop-by-hop toward ***dst***.

If an OvIP router r_{k-1} determines that the router on the next-hop toward ***dst*** does not support OvIP,[3] it does not forward a Route-Ping to that neighbor. Instead, it terminates, sending the Route-Ping datagram containing S(r_i, r_k) back to the initiator. Some policies that determine when to issue a Route-Ping are discussed in Section 8.

While the above is one sufficient method to generate OvIP source routes, other mechanisms that are not purely hop-by-hop could be used. For example, as with Tag Switching interior and exterior routing could be decoupled. This could be used to decrease Route-Ping overhead.

When a Route-Ping is to be forwarded between two border OvIP router/switches, the border entry router/switch could append to the source route all forwarding state needed to reach the border exit router/switch and subsequently, use IP tunneling to forward the Route-Ping directly to that border exit router/switch.

3. A neighboring router that does not implement OvIP will not respond to a Route-Ping message.

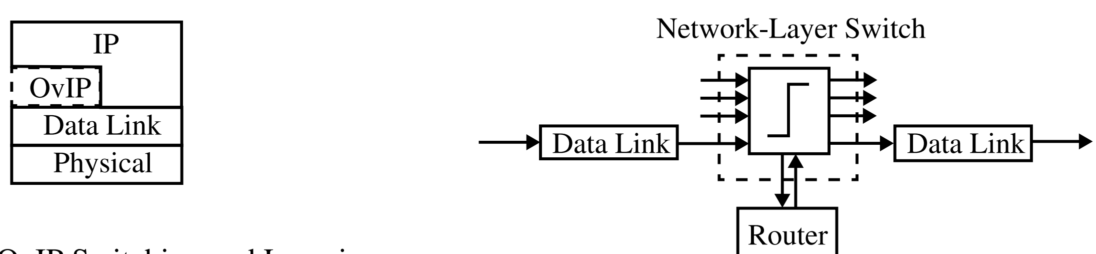

Figure 4. OvIP Switching and Layering

4 Initiator Encapsulation

Ideally, end users should not know that OvIP has been used to forward their datagrams [12]. Fragmentation is a concern because the OvIP datagram {OvIP header+payload datagram} is longer than the payload datagram alone. The MTU supported by an OvIP source route may be large enough to forward the payload datagram, but be too small to forward the OvIP datagram. Other transparency concerns are hop-count limits and hop-by-hop header options. In general, the presence of hop-by-hop header options in a datagram indicate that OvIP encapsulation not be used.

4.1 TTL and Hop Limit

The Time-To-Live in IPv4 and the Hop-Limit in IPv6 are easily accommodated. To observe these the initiator should subtract the hop-count of the OvIP source route from the hop-count limit field before forwarding. If the limit would be exceeded, appropriate action should be taken by the initiator. In IPv4 the header checksum must also be appropriately adjusted [13][14].

4.2 Fragmentation

Network-layer fragmentation in IPv6 is performed only by source hosts, while in IPv4 it may be performed by routers. Various mechanisms exist to address fragmentation that may be induced by OvIP encapsulation. In future, we expect sources to be engineered to take advantage of MTU information.

5 Source Route Stability

When a source route used to forward a datagram *d* differs from that path *d* would take if it were instead routed hop-by-hop, that source route is *stale*. A source route can go stale when a router along the source route makes a change to its routing table that affects *d*. The persistent use of a stale OvIP source route can produce significant negative outcomes:

- Breaking a connection
- Partitioning
- Stable routing loop
- Routing policy violation
- Use of a non-optimal path

Because OvIP source routes are derived by routers, periodically re-deriving a source route reflects changes made to the underlying routing state. An initiator r_i rederives a source route by reissuing the Route-Ping and utilizing the newer source route. Assume that r_i re-derives within a period $\mathbf{PR}_i$. This bounds the length of time that a stale source route is used to approximately $\mathbf{PR}_i$, as long as the round-trip time of the Route-Ping is small compared to $\mathbf{PR}_i$.

None of the first three negative outcomes above should occur for a connection forwarded across a source route $S(r_i, r_k)$, unless $\mathbf{PR}_i$ is too long. The last two negative outcomes are less severe because they would not result in a dropped connection and would tend to persist no longer than $\mathbf{PR}_i$. Two factors affect the choice of $\mathbf{PR}_i$. One is the typical interval $\mathbf{PC}_{ik}$ between router table changes across $S(r_i, r_k)$. The other is the connection time-out intervals of transport-layer protocols.

Time-outs of connection-oriented transport protocols are commonly set to between 25 seconds and a few minutes. A connection would not be dropped due to use of a stale source route if two conditions are met: (1) $\mathbf{PR}_i$ is much smaller than the connection time-out used by that connection's transport protocol, and (2) $\mathbf{PR}_i$ is much smaller than $\mathbf{PC}_{ik}$.

The first condition is met by setting $\mathbf{PR}_i$ substantially less than 25 seconds. To allow at least three refresh messages to be issued during a 25 second interval to account for occasional message loss, $\mathbf{PR}_i$ should not be much longer than eight seconds. We now examine the overall stability of end-to-end routes across the Internet to determine how reasonable the second condition that $\mathbf{PR}_i \ll \mathbf{PC}_{ik}$ is in practice. Cases where that condition fails will then be examined.

5.1 Persistence of Typical Routes

The persistence of end-to-end routes across the Internet has been the subject of a recent study by Paxson [16], who monitored routes across 8% of the approximately 1000 ASs that made up the Internet. This showed that at least 99% of all routes were stable for ten minutes or greater. Less than a percent varied on the order of a minute or less. Thus if $\mathbf{PR}_i$ is set to no longer than eight seconds, the condition $\mathbf{PR}_i \ll \mathbf{PC}_{ik}$ is easily true for at least 99% of end-to-end routes. If it is true for end-to-end routes, it is true for portions $S(r_i, r_k)$ of end-to-end routes.

5.2 Atypical Routes

There remains up to one percent of end-to-end routes that vary rapidly. Paxson identified these with "fluttering" and "deflection" policies undertaken by particular routers, and routing pathologies. Fluttering occurs when a router load-splits between two other routers, resulting in every other datagram sent to a particular destination travelling a different route. Deflection routing occurs when a router decides to send a datagram via a non-optimal path, either because of temporary congestion or to simplify a routing decision. Routers that adopt either of these policies could not generally use OvIP.

6 OvIP Overhead

There are three principal sources of overhead associated with OvIP. These are: (1) the cost of deriving and maintaining source routes, (2) added length of an OvIP encapsulated datagram, and (3) routing table space in initiators to hold source routes. Each of these is now discussed.

6.1 Re-derivation of Source Routes

Assume that the source route S(r_i, r_k) is associated with the destination ***dst***. The cost of sending a Route-Ping message is proportional to the cost of sending a datagram hop-by-hop along $r_i \ldots r_k$ and back. Re-derivation overhead is proportional to the fraction of Route-Ping refresh messages sent compared to the total number of datagrams forwarded toward ***dst*** per unit time, **F**(r_i, ***dst***), across S(r_i, r_k). That overhead is proportional to the inverse of $\mathbf{PR}_i * \mathbf{F}(r_i, dst)$. Assuming $\mathbf{PR}_i$ = 8 seconds, overhead falls rapidly as **F**(r_i, ***dst***) rises.

6.2 Encapsulation

A prototype packet format for OvIP is defined in Figure 1. The overhead to use OvIP is one octet plus the length of the source route. According to [16], the diameter of the Internet in 1996 is slightly greater than 30 hops. Assuming two octets per hop for the source route, a 30-hop source route would consume 60 octets, producing an initial overhead of 61 octets that declines to zero when forwarded by the terminator, since the source route is consumed during transit.

6.3 Routing Table Size

An OvIP source route is received by an initiator r_i in response to a Route-Ping. It takes the form <***dst***, S(r_i, r_k), MTU>. This augments the hop-by-hop routing table at r_i. Since initiation is optional, the size of an OvIP routing table is controlled by the initiator. Some initiation policies are discussed in Section 8.

An entry <***dst***, S(r_i, r_k), MTU> should be removed from an initiator's routing table if S(r_i, r_k) has not been used to forward to ***dst*** for several periods $\mathbf{PR}_i$.

7 Policy Concerns

Both virtual-circuit routing and source routing can be used to violate usage policy. It is possible to improperly use an OvIP source route, create an illicit source route or to snoop and copy one. The issue of policy violation is now discussed.

7.1 Source Route Authentication

Because the OvIP source route field is privately defined by each OvIP router/switch along the route, a router/switch can include in that field a value $\boldsymbol{f}(x)$ that is inverted by the switch to authenticate it. This facility prevents the casual illicit construction of source routes, but it would not prevent spoofing.

Because source routes are periodically re-derived, the value $\boldsymbol{f}(x)$ or the function could be changed over a longer period to make the illicit construction of usable source routes more difficult. This would require that new functions and values be communicated between the switch and its router.

7.2 Source Routing and Firewalls

An OvIP source route S(r_i, r_k) causes an encapsulated datagram to be switched past routers $r_{i+1} \ldots r_{k-1}$ without their receiving that datagram. Since the routing policies of $r_{i+1} \ldots r_{k-1}$ are then not consulted, it is possible for r_i to send a datagram to r_k via OvIP that would not have been forwarded to r_k if hop-by-hop routing were used.

Under normal circumstances this should not present a problem, because $r_{i+1} \ldots r_{k-1}$ each participate in creating S(r_i, r_k). Presumably, routers will not agree to create a path that violates their policy and if a policy change did occur along an active path, r_i will adjust to that new policy within $\mathbf{PR}_i$. However, it is possible that an illicit source route could be synthesized, that r_i could lie about the datagrams it intends to forward by putting the wrong datagram header into the Route-Ping message it creates, or that r_i holds onto and tries to use an old source route.

To prevent that abuse, it is sufficient for a firewall: (1) not to reply to any Route-Ping messages received over an untrusted interface, and (2) discard any OvIP datagrams arriving over an untrusted interface.

Another concern is that if a flow is allowed past a firewall, that the initiator may change the destination address in the payload datagram within the OvIP datagram sent over that flow and, by so doing, obtain access to a prohibited destination. IP Switching [2] prevents this by requiring that the router at the termination of a multi-hop virtual circuit store those header fields used when a circuit was constructed and subsequently, compare them with the actual headers received over that circuit and discard non-conforming datagrams. An OvIP terminator for which this is a concern may perform a similar operation.

8 Initiation Policies

The general question of when initiation should not occur is now discussed, followed by policies that govern when initiation should occur. Finally, the use of OvIP by end-system hosts is discussed.

8.1 Basic Policy Restriction

Assume that initiator r_i receives a datagram *d* sent to a destination for which it has not learned a source route. r_i performs a routing lookup to forward *d*. r_i must also decide whether or not to issue a Route-Ping for *d*. These events repeat along the sequence of routers $r_i \ldots r_{k-1}$. In general, Route-Pings should not be issued by $r_{i+1} \ldots r_{k-1}$ in response to receiving *d,* because they would be superfluous.

An OvIP router/switch that receives a datagram from a neighboring OvIP router/switch should not initiate. Presumably, an upstream router/switch initiated. For an AS comprised of OvIP router/switches, this policy restriction should be followed by interior routers. Border gateways may relax that policy, but should be aware that promiscuous initiation by proxy for a neighboring non-OvIP router could result in large OvIP routing tables. Policies that prevent unrestrained OvIP table growth are discussed below.

8.2 Intra-AS

An AS can exploit OvIP to improve interior forwarding performance by having each of the *n* routers in the AS initiate paths to each of the other routers in the AS. This produces an OvIP routing table whose expected size is proportional to the product of *n* and the mean hop-count across the AS.

This is analogous to permanent virtual circuits (PVCs) used to interconnect the routers in an ATM-based AS. Because the OvIP table space is strictly bounded and small, it is also provides a convenient method by which a border gateway can act as a proxy initiator to apply OvIP to external traffic that transits an AS.

8.3 Transit Traffic

If an AS allows external traffic to cross it, should it employ OvIP forwarding for that traffic? A border gateway has two principal cases to consider.

8.3.1 OvIP Traffic

OvIP transit datagrams will be switched along their source routes. Border gateways would not make OvIP routing table entries for this traffic.

8.3.2 Non-OvIP Traffic

Non-OvIP transit datagrams could be routed across the AS hop-by-hop. This would not take advantage of OvIP switching across the AS.

If OvIP is used, the border gateway should not promiscuously act as proxy initiator for all destinations seen in transit traffic. That could result in very large OvIP routing tables with attendant overhead.

A compromise assumes the intra-AS policy of Section 8.2 and forwards the traffic across the AS using the subset of intra-AS paths that reach other border gateways in the AS.

8.4 Local-Host Population

An OvIP router/switch could adopt a promiscuous proxy initiation policy for a population of local hosts that it serves. The number of destinations to which that population is sending traffic is presumably not so large as to preclude creating an OvIP routing table entry for each destination. In contrast to the paths used in Section 8.3.2, paths created for local traffic could extend beyond the boundary of the local AS if the neighboring AS supports OvIP. The effect of this policy and the policies discussed in Section 8.3 are illustrated in Figure 5.

A limit could also be placed on the number of locally generated OvIP routing entries. These could be managed by FCFS or LRU techniques.

8.5 Source Hosts

A source host could directly use OvIP assuming that its locally designated router supports OvIP. When source hosts use OvIP, proxy initiation by the local router/switch is not needed, since those hosts would initiate and maintain their own OvIP routing tables.

9 Prior Related Work

OvIP methodology combines source routing with network-layer switching to increase forwarding performance. Cheriton made a similar proposal [1]. IP over ATM and IP Switching [2] increase forwarding performance by taking advantage of ATM virtual-circuit support in the link layer to construct a circuit that corresponds to an IP flow and subsequently, switch that flow across an ATM cloud. Tag Switching [4] generalizes IP Switching to multiple link-layers.

9.1 Sirpent[4]

Sirpent [1] proposed replacing connectionless forwarding with source-routed, network-layer switching. A Sirpent source route is composed of a sequence of switch-specific forwarding directives. OvIP shares these characteristics with Sirpent.

OvIP's primary difference from Sirpent is that OvIP is designed to operate within the Internet infrastructure, whereas Sirpent utilizes a new hierarchy of route servers and it requires that source routes be bi-directionally symmetric.

4. Sirpent is a trademark of Stanford University.

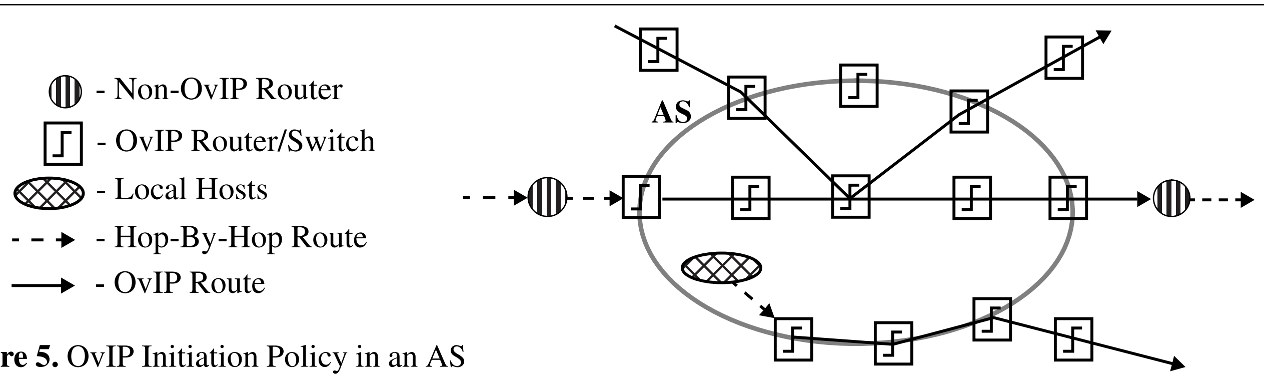

Figure 5. OvIP Initiation Policy in an AS

9.2 IP Switching Over ATM

The use of link-layer switching to route flows of IP datagrams across a cloud of ATM-based routers is perhaps best exemplified by the Ipsilon Networks IP Switch mechanism [2][3]. A cloud of IP Switches uses its underlying ATM link layer to create point-to-point links. This provides the basic hop-by-hop routing infrastructure needed to forward datagrams. However, multi-hop ATM virtual circuits can also be built.

An IP Switch gathers flow information by examining the datagrams that pass through it. A classification is made based upon the IP/TCP/UDP header contents. If it is judged that this datagram represents part of a flow, a virtual circuit and identifier (VPI/VCI) may be created for it and an advisory message sent upstream that allows subsequent datagrams that have the same header to be relabeled using the VPI/VCI created for this flow. These datagrams may now be switched across the ATM cloud.

There are two key functional differences between IP Switching over ATM and OvIP. First, OvIP forwarding state is cached only at the initiator, whereas ATM virtual-circuit state is distributed across all the switches that comprise that virtual circuit. Second, OvIP does not require ATM link-layer support.

9.3 Tag Switching

Tag Switching is a variation on IP Switching that is not restricted to ATM-only link-layers. Tag Switching associates a unique tag value with each distinct flow. The forwarding table in the Tag Switch is called a Tag Information Base (TIB).

The TIB is potentially large, since a separate TAG value is needed for each distinct flow. The key functional difference between Tag Switching and OvIP is that Tag Switching requires that flow forwarding state be distributed across all the switches that comprise a distinct flow. The number of distinct flows that can follow a particular path is limited to the size of the smallest TIB table along that path. Because OvIP flow forwarding state is maintained only at the head of a flow, the number of distinct flows that can pass through an OvIP switch is practically unbounded.

10 Further Work

This paper proposes source routing as an alternative to distributed-state switching protocols for forwarding IP traffic. However, while we have shown that OvIP methodology could be used in the Internet, this work is preliminary.

There are a number of performance and overhead issues that need to be examined through simulation and prototype deployment. These remain topics for future investigation.

11 Summary and Conclusion

This paper has investigated whether a network-layer switching methodology based upon source routing (OvIP) can successfully augment the hop-by-hop routing of connectionless datagrams. The forwarding state needed to route a datagram toward a destination across a sequence of routers can be gathered. That state constitutes a specialized source route that can be used to forward subsequent datagrams toward the same destination.

Source routing allows datagrams to be switched at the network layer, from incoming to outgoing links, where link-layers may be of differing technologies. Consequently, as switching replaces hop-by-hop routing, forwarding performance within the internetwork becomes less dependent on router performance.

12 Acknowledgments

The authors thank Joe Bannister, Mike Carlton, Ted Faber, Ramesh Govindan, John Heideman, Steve Hotz, Bill Manning, Jon Postel and Walt Prue.

13 References

[1] Cheriton, D. R.
Sirpent: A High-Performance
Internetworking Approach, SIGCOMM '89.

[2] Newman, P., Lyon, T., Minshall, G.
Flow Labelled IP: A Connectionless Approach to ATM
Proc. IEEE Infocom, San Francisco, March 1996.

[3] Newman, P., Edwards, W. L., Hinden, R., Hoffman, E., Ching Liaw, F., Lyon, T., Minshall, G.
RFC1953: Ipsilon Flow Management Protocol Specification for IPv4
May 1996.

[4] Rekhter, Y., Davie, B., Katz, D., Rosen, E., Swallow, G.
Tag Switching Architecture Overview (work in progress)
Cisco Systems, September 1996.

[5] Doolan, P., Davie, B., Katz, D., Rekhter, Y., Rosen, E.
Tag Distribution Protocol (work in progress)
Cisco Systems, March 1997.

[6] Postel, J., Reynolds, J.
RFC 1042: A Standard for the Transmission of IP Datagrams over IEEE 802 Networks
February 1988.

[7] Braden, R., Clark, D., Shenker, S.
RFC 1633: Integrated Services in the Internet Architecture: an Overview
June 1994.

[8] Deering, S., Hinden, R.
RFC 1883: Internet Protocol Version 6 (IPv6)
December 1995.

[9] Postel, J.
RFC 791: Internet Protocol
September 1981.

[10] Postel, J.
RFC 792: Internet Control Message Protocol
September 1981.

[11] Conta, A., Deering, S.
RFC 1885: Internet Control Message Protocol (ICMPv6)
September 1995.

[12] Woodburn, R., Mills, D.
RFC1241: A Scheme for an Internet Encapsulation Protocol: Version 1
July 1991.

[13] Braden, R. T., Borman, D. A., Partridge, C.
RFC1071: Computing the Internet Checksum
September 1988.

[14] Mallory, T., Kullberg, A.
Incremental Updating of the Internet Checksum
BBN Communications, January 1990.

[15] Mogul, J. C., Deering, S. E.
RFC 1191: Path MTU Discovery
November 1990.

[16] Paxson, V.
End-to-End Routing Behavior in the Internet
SIGCOMM '96.