A Virtualized Link Layer with Support for Indirection

Richard Gold and Per Gunningberg
Department of Information Technology
Uppsala University, Box 337
S–75105 Uppsala, Sweden
{firstname.lastname}@it.uu.se

Christian Tschudin
CS Dept, University of Basel
Bernoullistrasse 16
CH–4053 Basel, Switzerland
Christian.Tschudin@unibas.ch

ABSTRACT

The current Internet today hosts several extensions for indirection like Mobile IP, NAT, proxies, route selection and various network overlays. At the same time, user-controlled indirection mechanisms foreseen in the Internet architecture (e.g., loose source routing) cannot be used to implement these extensions. This is a consequence of the Internet's indirection semantics not being rich enough at some places and too rich at others. In order to achieve a more uniform handling of indirection we propose SelNet, a network architecture that is based on a virtualized link layer with explicit indirection support. Indirection in this context refers to user-controlled steering of packet flows through the network. We discuss the architectural implications of such a scheme and report on implementation progress.

General Terms

Design, Reliability

Keywords

Virtualized Link Layer, Indirection, Underlay Networks

1. INTRODUCTION

One of the requirements on the Internet today that was not part of the original design is that of indirection. Indirection enables such applications as mobility, proxies and route selection. Current solutions to these problems inside the Internet architecture include Mobile IP for mobility [11], application-specific solutions for proxies [3] and overlay networks for route selection [1]. These systems try to provide indirection by either extending the network layer or by building an application-layer network since the functionality that they require is not present in the current Internet. However each system is separate from the other and each requires deployment of separate infrastructure. Loose-source routing would have been one potential way to implement these services at the network layer but it has been disabled in most ISP networks for performance and security reasons because it was difficult to control and manage.

SelNet is a virtualized link layer (VLL), it sits at layer 2.5 and provides a completely flat topology with no structure to the network layer. It provides explicit indirection hooks to the network layer for control purposes. These indirection hooks occupy a similar space to ARP in IPv4 or Router Discovery in IPv6 in that they provide a mapping between a network layer address and a link layer address. Internally SelNet uses a label-switching approach for forwarding packets through the VLL. Due to its virtualization properties SelNet does not influence the topology or technology of the underlying link layer. It is based on the *network pointers* approach detailed in [13].

SelNet proposes a uniform approach to managing indirection. It resolves all address resolution activities into its own internal VLL representation. By requiring explicit resolution of addresses, SelNet becomes a natural point to handle mobility, route selection, NAT and other indirection requirements. Our contribution is the architectural principle that indirection should be implemented towards the bottom of the network stack since it is a generalizable principle applicable to many systems. We argue that a uniform approach to performing and managing indirection will help create more efficient indirection structures instead of multiple partially overlapping structures. Additionally, contrary to existing approaches, SelNet addresses *packet processing functions* in the network rather than nodes. Examples of packet processing functions are simple forwarding, multicast fan-out, media transcoding or compression. We note that there is significant benefit, in terms of flexibility and potential system lifetime, in not specifying how nodes are addressed, but rather leaving that to the protocol stacks which are hosted by SelNet.

1.1 Diagnosis

In the Internet a node is able to communicate with any other node in the Internet without prior invitation i.e., signaling. This unrestricted model of universal connectivity, although desirable for its simplicity and user empowerment, allows unsolicited communication and therefore opens up the way for Denial of Service (DoS) attacks. This connectivity model does not allow for per-application expressions of connectivity requirements. There is much debate about NAT and its site isolation principle[1] breaking the fundamental principles of the Internet, but it illustrates that there is a real desire for controlling or limiting universal connectivity. We believe that individual expressions of connectivity requirements will allow for a better balance between the users of the network and the network operators. For example, a video conferencing application that needs a certain port range open on a NAT box in order to receive incoming packets should be able to express these requirements. Furthermore, the Internet's transparency to applications in that they do not have any knowledge about the network in-between

[1]We note here that although address space reuse was the original goal of NAT, site isolation has become its most compelling feature.

them is widely held as favorable, but also limits the influence users can have on the path. In practice a default route entry is used. This causes difficulty for any application that wishes to take an indirect route to its destination.

The main cause of this current architectural stress is how the change in requirements which are being placed on the Internet affect the properties listed above. Mobility makes transparency harder to achieve since the network needs to be consulted about the current location of the end-point. Site isolation defies the principle of universal connectivity as it removes a section of the network from the globally addressable space. Additionally, the need to protect against distributed denial of service (DDoS) attacks make it undesirable to keep unsolicited communication as a fundamental principle. In order for the Internet to keep its phenomenal growth, it will have to change to satisfy these new requirements.

2. RELATED WORK

Related work in this area can be divided into two main areas: one is other underlay techniques, the second is indirection architectures.

2.1 Underlay Networks

Multi Protocol Label Switching (MPLS) is an underlay network which uses label switching into the network for faster and simpler packet forwarding. In order to achieve this goal a label is added to the packet when it enters an MPLS enabled network. This label identifies an action in the next hop, telling it how to forward the labeled packet. When the packet has reached the boundary of the MPLS enabled network the label is removed and regular IP routing is performed. IP forwarding is computationally more expensive than label switching since inexact matching (e.g., longest prefix matching) needs to be performed as opposed to exact matching for label switching.

SelNet, like MPLS, also introduces label switching between the network and link layer. This makes MPLS a natural target for internetworking with SelNet. In [16], we attempted to internetwork MPLS and SelNet so that an underlay path could be built which was partially an MPLS path and partially a SelNet path. IP traffic could then flow over this multi-hop path without any IP routing taking place. We discovered that the crucial difference between MPLS and SelNet is that MPLS labels must be distributed among all MPLS routers in the MPLS network whereas with SelNet the labels are local to each SelNet node, but we can also choose to globally assign some selectors although this is not mandatory. MPLS labels are used to address paths whereas the SelNet labels address functions. This allows us greater flexibility and control over packet processing since we can redirect SelNet packets to functions rather than just to nodes. This property is useful for extending the functionality of the network.

2.2 Indirection Architectures

The fact that the Internet suffers from being overly direct has also been observed by the Internet Indirection Infrastructure (i3) project [12]. In order to provide indirection they use a Rendezvous approach i.e., meeting at the middle, at their i3 servers. For looking up the appropriate server they use the Chord Distributed Hash Table (DHT) lookup service. A receiver puts a key called a trigger into the lookup service. This trigger is then used by the sender to route a packet through the i3 overlay network to the receiver. The goals of SelNet and i3 are very similar however the approaches are very different. i3 restricts itself to IP names as the only type of addresses and to IP forwarding as the single supported packet processing function. The i3 proposal is positioned architecturally as an overlay network i.e., above IP, whereas SelNet is an underlay network i.e., below IP. We see these two systems as being essentially complementary in nature since SelNet could be one mechanism used to redirect certain packets to the i3 service for further processing e.g., end-system multicast or a lookup service for tracking end host mobility.

Plutarch [5] is a network architecture proposal for bridging disjunct networking contexts to form a cohesive network. It is comprised of contexts which are groups of network elements (hosts, switches, routers etc.) that are homogeneous in terms of naming, addressing, routing and transport protocol. Contexts are bridged together by the use of interstitial functions (IFs) which are inserted between contexts to map between them. IFs are used to provide indirection by being able to chose which context to map a particular packet flow onto. Plutarch is intended to be a architecture which can express the heterogeneity of the current Internet as well as future networking systems by dividing them into homogeneous contexts. Plutarch and SelNet share a common approach of making the heterogeneity inherent in the Internet explicit and controllable. Plutarch does not specify mechanisms to actually perform this task, but rather leaves it to the actual implementation details of each particular context.

Active Names [15] is a network architecture that virtualizes the name resolution process in order to interpose new services. Active Names provides indirection through this virtualization since requests can be redirected to any arbitrary service. This system shares many similarities with SelNet, especially in the goals of the project. However, the architectural choices that are made are quite different. Active Names inserts itself into the name resolution activity (e.g., a DNS lookup) whereas SelNet inserts itself into the link layer address resolution activity (e.g., ARP). Since SelNet is designed to interface to network resolution activities, we need to sit below the network layer. Active Names, on the other hand, is concentrating on introducing extensibility into the name resolution process. Additionally, the Active Names system is invoked once per-connection which is acceptable for the applications that they specify, however we wish to be able to cope with more dynamic networks which may change their properties during their lifetime.

The Resilient Overlay Networks (RON) project [1] builds an overlay network on top of IP in order to get around the lack of loose source routing in the current IPv4 Internet. The general approach is to set up a group of RON nodes at various places in the Internet which form an overlay mesh over the Internet topology. The intent behind this is that when the default route through the Internet to a particular destination fails, there will exist an alternative route through the RON mesh since multiple providers will be providing the connectivity of the mesh. As a short term solution, RON is attractive and appears to work well, however, we agree with the authors of [6] that overlay approaches such as these will not work in the long term due to the complexity of adding additional control mechanisms on top of the network and the topology and capacity mismatch between the overlay and the underlying network. We advocate an underlay approach which attempts to strikes a balance between the needs of the users and the network operators.

3. SELNET

SelNet consists of two parts: XRP (eXtensible Resolution Protocol) and SAPF (Simple Active Packet Format). XRP is a uniform interface to access and configure SelNet's resources. SAPF is a minimal packet format providing basic demultiplexing functionality through the use of *selectors*[2]. In this section we show how this architecture works and how the associated "virtualized link layer" model permits us to embed different network personalities and to extend and redefine their services. Throughout this section of the paper, we assume that every node in the network is SelNet capable. We then discuss in section 4.1 a more realistic scenario where we show SelNet's ability to co-exist with the existing network infrastructure through partial deployment.

3.1 Static Forwarding in SelNet

SelNet positions itself as an underlay network, which means that it sits below the network layer and exports an indirection primitive to the upper layers of the protocol stack. As a virtual link layer, it allows us to use any available datagram service as a link layer to SelNet e.g., ethernet, or IP itself. In the following example we assume that we map the virtual link layer to a real ethernet thus reconstructing the current Internet model.

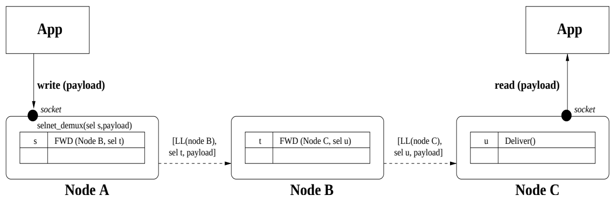

Figure 1: Forwarding in a static SelNet setup. (FWD= forwarding function, LL=link layer)

The simplest example of how SelNet functions is to describe how forwarding is done in a static environment. Packet forwarding inside SelNet's virtual link layer is based on "selectors": Each SelNet packet on the wire carries beside its link layer destination address an additional address field called selector. This packet format is defined by SAPF (Simple Active network Packet Format). The selector address, an flat 64-bit value, identifies the function which is to process an incoming packet (similar to a flow or path ID). The payload is some form of arbitrary content which is handled by whatever function is assigned to the selector in question. Selectors have different values depending on how they are assigned.

Figure 1 shows a scenario where selectors are static and preallocated. In this example an application on Node A wishes to communicate with an application on Node C. In order to do so, the application on Node A opens a socket to SelNet and writes to it, thus invoking the **selnet_demux** operation. When this operation is called, the forwarding function is invoked since the selector s, which is internally bound to the socket, is used in this example to communicate with the remote application. The forwarding function performs two tasks: it rewrites the selector from s to t since selector s is only valid inside Node A, and then sends the packet with the rewritten selector to Node B. Selector t on Node B corresponds to the forwarding function which will carry the packet over the next hop to Node C. Once again the selector is rewritten, this time from t to u.

When the packet reaches the destination i.e., Node C, selector u is demultiplexed and the payload is passed to the function which is associated with selector u. In this case, it is a local delivery function which passes the payload of the packet through a socket to the application. In the next section we show how dynamic resolution is used to establish selectors in nodes in the network.

Note that it is not necessary to rewrite applications to use selector sockets instead of IP. Because SelNet positions itself as a (virtual) link layer, an adaption layer can be inserted between the IP layer and SelNet which maps IP addresses to selectors in the same way as IP addresses are mapped to ethernet addresses. Thus, existing applications can still continue to access networking functionality via the IP sockets API although IP traffic is carried over SelNet.

3.2 Generalized ARP replacement

In the previous example we showed how SelNet works when selectors are statically assigned (e.g., by an authority such as IANA). Static, global selectors function only when all nodes independently agree on the assignment. This is because there is no distribution mechanism in SelNet for distributing labels as there is in MPLS, for example. However, selectors can also be dynamically assigned by communicating nodes since selectors only have validity on the node that assigns them.

In SelNet, dynamic forwarding state can be set up in a way similar to ARP. First, the source node puts a query on a link which asks about how to reach a given target address. Then the target node will reply with its link layer address as well as a selector which identifies the requested target. In SelNet, this ARP style resolution is carried inside an "eXtensible Resolution Protocol" (XRP) packet format, which is the standard signaling means of SelNet.

More specifically, if a node A wants to communicate with node B, node A broadcasts a resolution request (RREQ) to the "well known" XRP selector. This request specifies the address that node A wishes to resolve (e.g., the IP address of B) and how the resolution should be done. This could be a ARP-style resolution, which would return a complete link layer + selector address pair, or a DNS-style resolution where only a translation between name spaces occur (e.g., from logical name to IP address), or a combined resolution (i.e., from logical name to link layer + selector address pair).

Figure 2 shows this process in more detail. Before sending out a RREQ, Node A installs a function at a new selector to handle any resolution reply. This function is identified by the selector 'r' in figure 2. Let's assume that B is the node that A wants to reach. B receives the RREQ (carrying the target address as well as A's reply details) **(1)**, it decides that it should send back a reply and prepares itself for receiving IP packets from A. To this end, node B creates a local selector entry 'd' (either randomly assigned or by other means) for the delivery of packets to its IP stack. The combination of B's link layer address eth_b and the selector 'd' is then sent back in a RREP to A's link layer address and selector 'r' **(2)**. Based on the RREP, node A will install a forwarding entry with selector 'f' pointing to the pair eth_b+'d' **(3)**.

Once this resolution step has completed, the system is now at the state where forwarding can take place as described in section 3.1.

3.3 Indirection via Multi-hop Resolution

The examples above show how forwarding is performed in SelNet and how state can be dynamically installed on nodes in the SelNet network. To create indirection, we combine these two processes to perform multi-hop resolution dynamically and thereby forward packets via an intermediate node. This process allows SelNet to setup a custom forwarding path through the network similar to a loose source route enabling potential indirection. With this forwarding path the sender does not know the location of the destination, it is abstracted away behind the selector that the sender uses to get to the next hop on the path.

[2]Thus giving the name "Selector Network" to SelNet

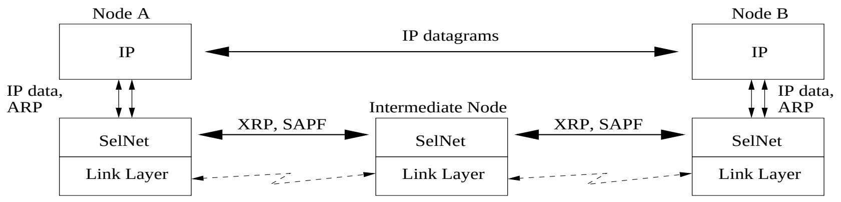

Figure 3: Dynamic Forwarding with SelNet

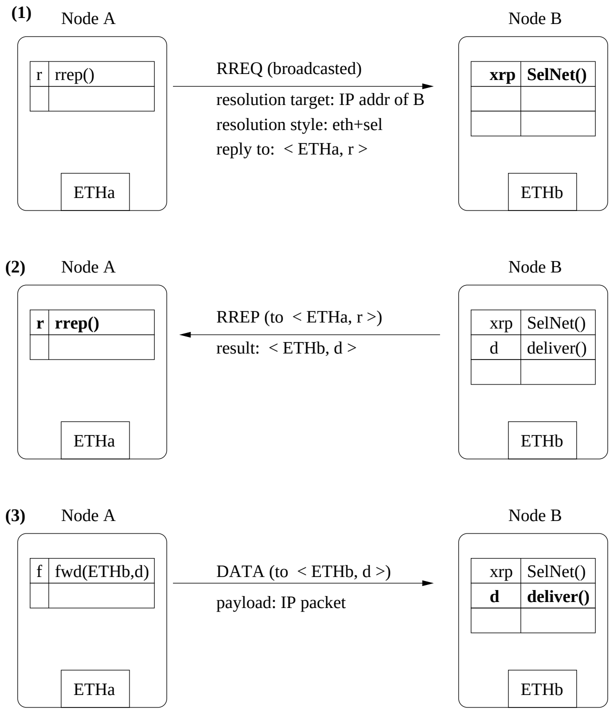

Figure 2: SelNet performing an ARP-style resolution.

We have implemented an ad-hoc routing protocol called *LUNAR* using SelNet for forwarding and demultiplexing which uses exactly this approach [14]. The corresponding configuration can be seen in figure 3. For the sake of brevity we do not include all the details of the XRP message contents. When the intermediate node between Node A & Node B receives A's RREQ, the address to resolve is checked if it matches the intermediate node's address. If this is not the case, then the request is routed towards the destination. How this routing decision is made is not specified by SelNet since routing is not part of the link layer of the protocol stack. It could be made by consulting IP routing tables, a lookup to a Peer-to-Peer infrastructure or, in the case of LUNAR, simply rebroadcasting to reach nodes outside of the source node's radio range. The resolution process can be viewed as the recursive application of the resolution process detailed section 3.2. Once the forwarded RREQ reaches Node B, the destination is checked to see if it matches Node B's address. If the match is successful, Node B sends a RREP to the forwarding function on the intermediate node. The RREP contains a selector pointing to Node B. Any packet sent from the intermediate node to that selector will end up at Node B and be demultiplexed up to the appropriate function. This is the same forwarding technique detailed in section 3.1. In this example, as in the previous sections, that function is the IP stack. The intermediate node then sends a RREP to Node A which contains a selector pointing to the intermediate node. When a packet is sent to that selector it is forwarded to the intermediate node and demultiplexed up to the function which will forward the packet to the network selector pointing to Node B.

The SelNet's resolution request/reply model allows *late binding* to be performed on which addressing schemes are supported by the network. This late binding can also be used to handle mobility as a resolution step needs to be performed before communication, thus ensuring that the most current version of the address mappings are discovered prior to communication. This removes the need for a separate system such as Mobile IP to handle mobility as the underlying SelNet infrastructure will not be as direct and transparent as the current Internet architecture.

3.4 NAT

NAT is a very difficult entity for the current Internet architecture to cope with as it breaks the fundamental principles of transparency and universal connectivity. However it does provide site isolation in a very cost-efficient way by not requiring the network or end systems to change. SelNet is able to provide site isolation functionality through the XRP resolution process. However, due to the indirection hooks that SelNet has, we are also able to selectively restore symmetric connectivity for those nodes behind the NAT which are explicitly authorized.

In order for a SelNet node inside of an isolated realm to be externally reachable, it must send an XRP message to a SelNet waypoint node (that replaces the NAT), requesting visibility. The waypoint node has to approve this XRP request before the appropriate entries are added to a name map. This name map depends upon which naming schemes are being run on top of SelNet. When a node outside of the isolated realm wishes to contact a node inside the realm, the corresponding resolution request will reach the waypoint. The request, if approved by the waypoint policy, will be forwarded to the potential destination. The destination can also choose to grant the resolution request or to deny it. A reply to the waypoint permits to either install the appropriate forwarding state on the waypoint, or to send back an error code in the XRP reply to the source node. This process allows symmetric connectivity to be selectively enabled, whilst keeping the power of veto with the waypoint node (that replaces the NAT).

3.5 Route Selection & Scalability

Since SelNet's selectors are opaque i.e., it is not known from the selector value itself what semantics are bound to it, we can attach different functions to selectors. In this section we show how selectors can be used to identify long-lived and public forwarding paths e.g., a transatlantic link or autonomous systems, rather than only local, small-scale network paths.

Resolution of delivery paths triggered by end users will not scale if these requests require discovery and resolution over the full path across the entire network. For scaling reasons, some form of aggregation will be needed. This implies a reduction of information complexity from a large number of intermediate nodes constituting a path to a single identifier representing a path through the network. One of the effects of this is that if a handful of paths exist between a source and a destination exist, it becomes feasible to allow end users to select one of these paths through the network for their traffic. This is analogous to Least Cost Routing consumer devices in the telecoms world which chose a provider to route over depending on cost.

The SelNet model for core nodes in the network is to make them as simple as possible i.e., to function as layer 2.5 switches which are only forwarding packets. We imagine various routing protocols running on top of SelNet and feeding SelNet with the appropriate information that it needs to set up paths. We agree with the authors of [7] that physically separating the route computation process from the routers themselves is a good strategy for reducing router complexity.

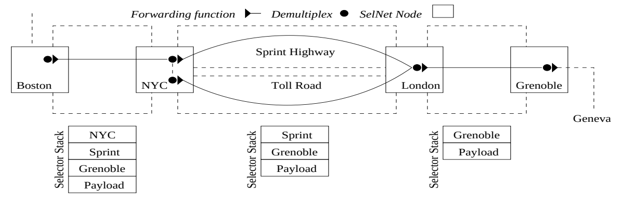

Figure 4: Highway routing with SelNet. Selector Stacks shown with attached payload.

3.5.1 Mapping a Destination to a Selector Stack

A first example involves an equivalent to autonomous systems on which IP routing is based today. Like MPLS, we envisage label stacks (selector stacks) inside packets. Instead of mapping a destination address to the next-hop autonomous system, SelNet can map a destination to a selector stack which describes a loose source route through the network. The process works as follows: When a source node requests the resolution of a destination, it will obtain a selector that points to a special local forwarding function. This forwarding function will add the selector which points to a routing function on the next-hop node. Additionally each packet sent to this function will be augmented by a selector pointing to the resolved loose source route. In other words, The front-most selector identifies a resolved path to the routing function which, instead of doing label switching, will pop the front-most selector from the selector stack. The second selector will then be demultiplexed into the node's SAPF table to invoke the function that will handle this packet. Typically in the core of the network this will be a simple forwarding function. The second selector functions as a *path* selector similar to PathIDs in BANANAS [9].

Consider now the case where each selector in this selector stack representing a loose source route maps to an autonomous system. This means that forwarding such a packet consists of taking the next selector from the stack and treat it as an entry point to a path to the named autonomous system. This path will be labeled with a statically assigned selector such as those discussed in section 3.1. Because connectivity at the AS level is long lived, we can proactively install such delivery paths and share them between many users. The difference to the IP case is that for IP, the mapping from IP destination number to AS number is implicitly done at each ingress point, while here this mapping is exposed to the end nodes and thus becomes "selectable". This allows the potential for the end-users to chose one of a set of paths through the network.

3.5.2 Mapping a Destination to a Data Highway

In the previous example we used selectors to identify trans-AS paths. Another routing abstraction, which extends the loose source routing concept, consists in using selectors to work with multi-exit *data highways*. Instead of resolving a destination to a path (identified by an entry selector), or a series of paths with mandatory waypoints (identified by a stack of selectors), we resolve a destination to a "highway-name + exit-name" pair. The highway name (selector value) specifies which route, among several possible ones if several providers exist, should be used to reach the destination. The exit name, represented by another selector value, specifies which exit on that highway to take. The exit identified in this way roughly corresponds to the waypoints that a packet will have to traverse. The advantage of this approach is that it provides a level of abstraction which the providers can use to dynamically change their network topology without affecting the traffic flowing through their network. As long as the exit-names are kept in the same order, it does not matter how the network is actually structured.

For example, there might be different transatlantic highways starting from New York to Geneva passing through London and having an exit at Grenoble. A user in Boston who wants to send a packet to Grenoble has several options now. In one case she could compose a packet with a source route "Boston/Sprint-Highway(NY,Geneva)+exit at Grenoble/default Grenoble router" and issue two resolution requests: one "name lookup RREQ" for the selectors to use for the highway and the exit points, and one RREQ to get a path to Boston. The latter selector permits to reach Boston, the former selectors to build a selector stack. In the Boston node, a packet's front selector would be popped, which diverts the packet to the transatlantic highway; Each intermediate node on this highway will check whether it matches the exit name. The Grenoble node will match and pop again a selector, finding that it has to be forwarded to the default local delivery function. This process can be seen in figure 4.

Let us further assume that the operator of this highway establishes a shortcut from London to Grenoble e.g., to counter some flash crowd problems. In this case, the packet would already by picked out of the data stream in London and reach Grenoble by this temporary bypass. Thus, the use of selectors offers traffic engineering flexibility on both sides.

Note that except resolution requests at the edge (Boston and possibly also at Grenoble), no end user state had to be stored inside the core network. Because highway and exit points are long-lived, their selector would be cached at the edge and corresponding RREQs would not need to go transatlantic. Note also that the only user-specific packet processing inside the core relates to fast selector stack inspections and forwarding decisions.

4. DISCUSSION

Here we discuss some of the architectural consequences and issues that arise from the design choices of SelNet.

4.1 Deployment

During the discussion in this paper, we have assumed a network where all nodes are running SelNet. This was deliberately chosen to show SelNet's capability as a stand-alone network architecture.

Since deployment on the current Internet is an extremely slow process, we discuss here how SelNet's underlay approach allows for incremental deployment.

One typical way of running SelNet is to use it as an underlay to IP. This means that any IP traffic sent from a SelNet node has the SAPF header inserted between the IP header and the Ethernet (or other link layer) header. The SAPF header allows the packet to be demultiplexed to the correct SelNet packet processing context on arrival. This method of deployment clearly positions SelNet as a virtualized link layer.

However, it is very likely that we would like to start SelNet deployment only in a handful of local area networks. These LANs may well be geographically distributed, but connected via the Internet. In which case, tunneling SelNet traffic over the Internet is one way of achieving SelNet connectivity between these LANs. Using an approach similar to Minimal IP Encapsulation we can insert the SAPF header between the transport and network layer headers so that SelNet traffic can be a) effectively routed through the Internet and b) correctly demultiplexed at the receiving node. Our current implementation of SelNet runs over both Ethernet and UDP. We chose UDP for implementation purpose rather than IP for speed of prototyping. Despite the additional overhead incurred from using tunnels, in our experience SelNet is still usable. This mix of both underlay and overlay approaches shows a promising direction for pragmatic future deployment.

4.2 Resilience

Since SelNet introduces some forwarding state on intermediate nodes, it must have some resilience properties to ensure recovery in the event of failure. Consider the example in figure 3, once the multi-hop path has been established SelNet will periodically refresh the selectors using a soft-state approach. Usually applications will do this matching the time cycles of ARP (120 seconds between refreshes). However, this can be configured on a per-application basis.

Referring back to figure 3, if Node A fails then the selectors present at both Node B and the intermediate node will be garbage collected after a certain timeout. Once the Node A comes back online, the selectors will not be present at the intermediate node, so Node A will have to perform the resolution again in order to ensure that a path still exists between it and its destination.

Whilst such an approach works for small networks, if SelNet is to scale up, then there must be some way of performing soft state recovery so that the maintenance load of XRP does not get out of hand. Inspired by the work done in [4] and [8] we propose a mechanism which uses reliable triggers (i.e., the full set of forwarding state) hop-by-hop to install state and then an end-to-end soft state refresh mechanism to ensure that these triggers are kept up-to-date and are re-installed correctly in advent of failure. This means that the full transmission of state happens at the beginning of the connection and in the case of failure. State is kept up-to-date by periodic refreshing. Thus the transmission of the full set of state is infrequent so the load is not too great, but the state is kept consistent by periodic refreshing by more lightweight mechanisms.

4.3 Security considerations

We outline in this section how access control and DDoS protection are performed in SelNet. Traditional measures of privacy, integrity and authentication are dealt with by standard cryptosystem approaches.

Typically, solutions that introduce more flexibility into the network are rightly considered as security risks as they provide a hook into the system which could be used to break in or take down the system. One such system was the original IP loose source routing which was usually switched off by ISP networks due to security, performance and business concerns. There are two reasons why we feel that SelNet is a better solution for network operators than loose source routing: firstly, it requires *explicit* agreement from the operator that a certain route is allowed to be taken and secondly, the network operator can choose exactly which routes are exposed to the users and which are not. These bilateral agreements are enforced by the XRP access control mechanisms detailed in the next section.

4.3.1 Access control via XRP

Access control is an important part of the SelNet architecture. By providing fine-grained access control to the user and network operator, we hope that more services will be introduced in a secure way. This is in contrast to the unsolicited communication principle that underlies the Internet architecture. We share the view of the authors of [2] who see security as an enabler of new functionality rather than constricting it. Access control is provided in SelNet through XRP. In SelNet no communication can take place until resolution is complete. Therefore it is much easier for a communication instance to be approved by intermediate nodes (typically an XRP policy box). In the Internet architecture, any node can send to any other node without prior agreement. This makes security difficult as legitimate communication is hard to distinguish from a malicious attack. Firewalls are an attempt to introduce some access control into the network, but they often cause problems by inadvertently blocking legitimate communication due to their lack of ability to discriminate between permitted and non-permitted traffic.

In XRP intermediate nodes have to approve both the destination and method of reaching destination. That is, before a selector can be sent back to the destination, the XRP resolution request has to be processed. These approval policies can be complex or trivial. For example, route all IPv4 traffic by default or route only traffic which conforms to a policy. The key point here is: Who is allowed to do what indirection? By allowing the approval policies to be calibrated as required, we can have access control suitable for certain areas of the network. For example, the core transit networks will very likely have very simple policies due to processing constraints whereas the edge or access networks will likely have more complex policies to match their application complexity.

4.3.2 Protection against DDoS

Since unsolicited communication is permitted by the Internet architecture, a node cannot express which packets it wishes to receive and which is does not. In [10] the authors also note this problem with the current architecture. We believe that SelNet can help with these DDoS attacks because an XRP resolution request would be sent to the nearby XRP processing box and that is the limit to where the DDoS attack could go without explicit authorization of the XRP box. Even if the local box is being DDoS'd, it is a simpler problem to handle than a DDoS which traverses multiple IP networks which obscures the identity of the original sender (especially if the packets are spoofed).

One proposal to counter DDoS attacks is Pushback. Pushback is a way of installing filters on intermediate nodes in the network in order to block certain address ranges which are known to be DDoSing other machines. SelNet, through XRP, provides a mechanism with which an end node can instruct intermediate nodes (for example, firewall or edge router) that certain types of packets should be blocked. Due to the extensible nature of XRP, we can easily deploy additional indirection schemes. We note here that this indirection scheme redirects certain packets into a black hole rather than to another packet processing function.

5. CONCLUSIONS AND OUTLOOK

The contribution of the SelNet network architecture is twofold: one is the architectural approach of placing indirection at the bottom of the protocol stack and the second is how to maintain flexibility in the face of changing requirements which SelNet aims to achieve by the usage of explicit resolution mechanisms and addressing packet processing functions rather than nodes.

The main advantages of SelNet are flexibility, performance and security: Being an underlay network also means that we are not constrained by the network layer stack like an overlay network. Since the addressable units of our architecture are functions rather than nodes, we allow for greater flexibility and extensibility than architectures which mandate an addressing mechanism for nodes. With label switching, packet forwarding is faster than with IP [17]. and we do not incur the packet header overhead of an overlay approach. Since SelNet is underneath the network layer, it is easier to secure SelNet since resolution is an explicit activity that must take place before data forwarding on the network layer can take place.

We have the core infrastructure of SelNet implemented and we have a real-world implementation of a simple distributed proxy scenario which we will continue to develop. Additionally we have a version of SelNet based on LUNAR implemented for NS-2 and one element of future work is to investigate user-controlled route selection and the general scalability of SelNet.

6. REFERENCES

[1] ANDERSEN, D. G., BALAKRISHNAN, H., KAASHOEK, M. F., AND MORRIS, R. Resilient Overlay Networks. In *18th ACM Symposium on Operating Systems Principles (SOSP)* (2001).

[2] ANDERSON, T., ROSCOE, T., AND WETHERALL, D. Preventing Internet Denial-of-Service with Capabilities. In *ACM Workshop on Hot Topics in Networking (HotNets-II)* (2003).

[3] ARDON, S., GUNNINGBERG, P., LANDFELDT, B., ISMAILOV, Y., PORTMANN, M., AND SENEVIRATNE, A. MARCH: a distributed content adaptation architecture. *International Journal of Communication Systems, Special Issue: Wireless Access to the Global Internet: Mobile Radio Networks and Satellite Systems* (2003).

[4] BERGER, L., GAN, D., SWALLOW, G., PAN, P., TOMMASI, F., AND MOLENDINI, S. RSVP Refresh Overhead Reduction Extensions, 2001. RFC 2961.

[5] CROWCROFT, J., HAND, S., MORTIER, R., ROSCOE, T., AND WARFIELD, A. Plutarch: An Argument for Network Pluralism. In *ACM SIGCOMM Workshop on Future Directions in Network Architecture (FDNA)* (2003).

[6] CROWCROFT, J., HAND, S., MORTIER, R., ROSCOE, T., AND WARFIELD, A. QoS's Downfall: At the bottom, or not at all! In *ACM SIGCOMM Workshop on Revisiting IP QoS (RIPQOS)* (2003).

[7] FEAMSTER, N., BALAKRISHNAN, H., REXFORD, J., SHAIKH, A., AND VAN DER MERWE, J. The Case for Separating Routing from Routers. In *ACM SIGCOMM Workshop on Future Directions in Network Architecture (FDNA)* (2004).

[8] JI, P., GE, Z., KUROSE, J., AND TOWSLEY, D. A Comparison of Hard-state and Soft-state Signalling Protocols. In *ACM SIGCOMM* (2003).

[9] KAUR, H. T., KALYANARAMAN, S., WEISS, A., KANWAR, S., AND GANDHI, A. BANANAS: An Evolutionary Framework for Explicit and Multipath Routing in the Internet. In *ACM SIGCOMM Workshop on Future Directions in Network Architecture (FDNA)* (2003).

[10] LAKSHMINARAYANAN, K., ADKINS, D., PERRIG, A., AND STOICA, I. Taming IP Packet Flooding Attacks. In *ACM Workshop on Hot Topics in Networking (HotNets-II)* (2003).

[11] PERKINS, C. IP Mobility Support for IPv4, 2002. RFC 3344.

[12] STOICA, I., ADKINS, D., ZHUANG, S., SHENKER, S., AND SURANA, S. Internet Indirection Infrastructure. In *ACM SIGCOMM* (2002).

[13] TSCHUDIN, C., AND GOLD, R. Network Pointers. In *ACM Workshop on Hot Topics in Networking (HotNets-I)* (2002).

[14] TSCHUDIN, C., GOLD, R., RENSFELT, O., AND WIBLING, O. LUNAR: a Lightweight Underlay Network Ad-hoc Routing Protocol and Implementation. In *Next Generation Teletraffic and Wired/Wireless Advanced Networking (NEW2AN'04)* (2004).

[15] VAHDAT, A., DAHLIN, M., ANDERSON, T., AND AGGARWAL, A. Active names: Flexible location and transport of wide-area resources. In *Usenix USITS'99* (1999).

[16] WESTLING, A. Internetworking MPLS and SelNet. Tech. Rep. 2004-013, Uppsala University, 2004. `http://www.it.uu.se/research/reports/2004-013/2004-013-nc.pdf`.

[17] WOLF, T., DECASPER, D., AND TSCHUDIN, C. Tags for high performance active networks. In *The Third IEEE Conference on Open Architectures and Network Programming (OpenArch)* (2000).