An Indirection Architecture for the Internet

Richard Gold

October 20, 2005
Contents

1 Introduction 1
  1.1 Background ................................................................. 2
  1.2 Problem Areas .............................................................. 4
    1.2.1 Deployment ............................................................ 4
    1.2.2 Collapsing Naming and Addressing ................................. 4
    1.2.3 Interconnecting multiple types of networks ...................... 6
  1.3 Thesis Research Areas ................................................... 7
  1.4 Contributions ............................................................. 8
    1.4.1 Implementation Work ............................................... 9

2 Related Work 11
  2.1 Introduction .............................................................. 11
  2.2 Underlay Networks ....................................................... 11
    2.2.1 MPLS ................................................................. 11
  2.3 Network Architectures .................................................. 12
    2.3.1 Bananas .............................................................. 12
    2.3.2 TRIAD ................................................................. 13
    2.3.3 Role Based Architecture ......................................... 13
    2.3.4 Metanet ............................................................... 14
    2.3.5 Yellow Book ........................................................ 14
    2.3.6 Active Names ....................................................... 15
    2.3.7 IPNL ................................................................. 15
    2.3.8 Nimrod ............................................................... 16
    2.3.9 Plutarch ............................................................. 16
  2.4 Overlay Networks ........................................................ 17
    2.4.1 Internet Indirection Infrastructure ................................ 17
    2.4.2 Layered Naming Architecture .................................... 17
    2.4.3 Resilient Overlay Networks ...................................... 18

3 Summaries of the Papers 19
  3.1 Paper A: Network Pointers ............................................. 19
  3.2 Paper B: SelNet: A Virtualized Link Layer ......................... 20
  3.3 Paper C: LUNAR: Lightweight Underlay Network Ad-hoc Routing
    Protocol and Implementation ........................................... 21
List of Figures

1.1 Before shows the classic passive model of protocol processing, after shows the new active model. Diagram from Christian Tschudin... 3
1.2 Meta-Architecture................................. 8
Chapter 1

Introduction

This thesis is concerned with how to extend the functionality of the Internet. The Internet of today is a vast network of inter-connected networks whose design dates back to a US Department of Defense project in the 1960s called ARPANET. In forty years, the Internet has evolved from the research laboratory to a critical piece of the global communications infrastructure. The Internet is a foundation, supporting the communication of many applications such as the World Wide Web (WWW), Email, instant messaging, multimedia and peer-to-peer file-sharing. These applications are used by millions of people all over the world. The Internet supports these applications by giving them the ability to send a piece of data (called a packet) to any computer or device which is connected to the Internet, irrespective of the physical location of the recipient. This service that the Internet provides to application is called best effort packet delivery. This means that the Internet will try to deliver a packet to a recipient, but that it makes no guarantees. Whilst the service itself is very simple, it is also very powerful. As the Internet of today has already shown, it is possible to support many different types of applications using this simple service. This service is the forwarding component of the Internet. Additionally the Internet needs to be able to pinpoint the recipient of the packet amongst the hundreds of thousands of networks that comprise the Internet. This process is complex and is the domain of the routing component of the Internet.

The goal of this thesis is to examine how to extend the forwarding and routing components of the Internet in order to achieve greater flexibility. As the Internet has grown, it has become increasingly hard to change it. On the 1st January 1983, the ARPANET changed its core protocol suite from NCP (Network Control Protocol) to TCP/IP thus marking the beginning of the Internet as we know it today. This was called the “Flag Day”. On this day, it was attempted to convert all machines on the ARPANET to TCP/IP. There were network outages during that day, but the network was working after a fashion the next day. The number of nodes that needed to be converted was well under 1,000. Now that the Internet is globally so important, it is

---

1A note on language: Richard Dawkins says this much better than I can: “I may refer to the ‘reader’ or ‘user’ as ‘he’, but I no more think of my readers as specifically male than a French speaker thinks of a table as female”. (refx!)
not feasible to have a day of downtime. Also it would be impossible to co-ordinate the 350,000,000 or so machines on the Internet. The growth of the Internet has therefore led to a corresponding decrease in its flexibility.

The obvious question to ask at this stage is: “why do we need flexibility?” If the Internet serves our purposes well enough, then the additional overhead incurred by introducing more flexibility into the system is not worthwhile. Our motivation for wishing to introduce more flexibility into the Internet stems from the observation that there is an increasing amount of architectural stress inside of the Internet. By this we mean that recent developments in the Internet are breaking or bending the fundamental principles that make the Internet what it is. We present one example here, but discuss more examples in section 1.2. One of the fundamental principles of the Internet is symmetric connectivity. Any host on the Internet should be able to communicate with any other host on the Internet. A typical example of this is the FTP protocol where a client connects to a server and then the server connects back to the client to continue the transaction. In order for this transaction to function correctly symmetric connectivity has to be present. In the Internet of today there are many devices such as NATs, Firewalls and Proxies which break this principle of symmetric connectivity, typically in the name of security. These actions go to show that the needs of the network users and operators are not being sufficiently met by the existing network architecture. The security restrictions which are deployed mean that certain applications such as Voice over IP (VoIP), online gaming and peer-to-peer filesharing have problems functioning as they assume symmetric connectivity. We believe that if the Internet was more flexible e.g., better able to specify and express connectivity requirements, then we would be better able to manage the conflicting concerns between the network and the applications that use it.

1.1 Background

The work presented in this thesis can be seen as a reaction to a trend in the field of computer networking. During the 1990s, many mechanisms were proposed by the research community to add functionality to the Internet. For example:

**IPv6**: The flagship replacement for IP version 4. Although the deployment of IPv6 is still active, it has been an process which has been ongoing for over six years at the time of writing.

**Multicast**: Group communication for the Internet is still not fully deployed. Whilst multicast support is present in some regions of the Internet, there are still issues with charging for multicast traffic that crosses provider domains (refx).

**Quality of Service (QoS)**: QoS is an attempt to bring service differentiation to the Internet in the style of traditional telephony. It enjoys some success in closed networks, for example TV-on-demand where one company operates and controls a dedicated piece of infrastructure. However it has not been widely deployed in interdomain scenarios.
1.1. BACKGROUND

These three examples have not been widely deployed for a variety of reasons. For example: requiring that every router in the Internet has to be upgraded, the protocols themselves being overly complex, lack of understanding of economic implications (e.g., Multicast charging and, most importantly, the immense difficulty involved with attempting to change wholesale a critical piece of the global communications infrastructure. A general frustration grew in the research community concerning these barriers to deployment of new Internet protocols and many looked for alternative ways to get their systems deployed. One approach was quite radical: Active Networking. This was an idea which in its purest form centered around making every router in the Internet a Universal Turing Machine and each packet a Turing Machine. This approach can be seen in figure 1.1. Active Networking deals with the lack of flexibility and the difficulty of deployment by every single packet containing the code that it needed in order to be processed correctly. So each packet could, potentially, carry its own application, transport and network layer protocols. Obviously, such an approach would be unworkable with current technology due to security and resource consumption issues, but as a concept it is intriguing. It would allow an unprecedented degree of customization of computer communication and thus remove the barriers to deployment that currently exist to new technologies.

![Diagram showing the classic passive model of protocol processing and the new active model](attachment:diagram.png)

Figure 1.1: Before shows the classic passive model of protocol processing, after shows the new active model. Diagram from Christian Tschudin.

Active Networking was one approach to solving the deployment problem. In retrospect it was probably an idea ahead of its time since many of the problems associated with Active Networking are due to limitations in other areas e.g., security and resource control. Using Active Networking as a departure point, researchers began to look at *indirection* which is the topic of this thesis. The reasoning behind indirection is to provide the means whereby a suite of functions in the network can be used. Instead of just a single set of functionality being available e.g., IPv4 packet forwarding, having some indirection functionality would allow packets to be redirected away from this default processing path and towards a processing path which performs a different task. Examples of different tasks include IPv6, transcoding, compression or customized routing decisions. These tasks are currently difficult to perform in the Internet. How to perform indirection and which functions should be available in the network are questions which are examined in detail in this thesis.
1.2 Problem Areas

From the issues described above, we extracted three main problems to work on. We examine the issues of deployment, the collapsing of identity, access and location and the issues involved in connecting different types of networks to the Internet.

1.2.1 Deployment

Invariants present in the Internet today i.e., the IPv4 address structure, are the main cause of the architecture’s lack of flexibility which leads to deployment issues [1]. There is no way to change the semantics or structure of the IPv4 address without breaking backwards compatibility. In order to extend the Internet’s functionality at the network layer, any proposal must inter-operate with legacy IPv4 routers otherwise deployment becomes, for all practical purposes, infeasible.

This lack of extensibility of the Internet is a theme running through much of the work in this thesis. All of the papers included involve the introduction of functionality which is not part of the current Internet architecture. There have been certain mechanisms present in the Internet to allow extensibility, such as IP options and Loose Source Routing. However, IP options are seen as extremely inefficient and are liable to be dropped by many routers (refx: pierre & andreas). The IP options approach also does not let us replace the existing IP layer easily, which is something we would like to do to enable IPv6 amongst other new protocols. Loose Source Routing has been switched off by most ISPs citing security and business concerns. Irrespective of the wisdom of these actions, there currently exists no way to extend the current Internet architecture in a way that does not introduce more brittleness. Whilst Overlay Networks are an excellent way of quickly deploying a new network architecture, they are not native to the existing infrastructure. They build their own network on top of the Internet by constructing tunnels. Therefore it is difficult to change the Internet itself with this approach. Overlays are useful for adding to the existing Internet architecture but not necessarily useful for changing it. They also suffer from their own class of problems such as brittleness by forcing the lower layers of the protocol stack to remain as they are and route stretch caused by the loss of locality information between the overlay network and the real network and also the introduction of more control loops into the network [6].

1.2.2 Collapsing Naming and Addressing

Shoch (refx) makes the following definitions which are now in common use:

- A name identifies what you want.
- An address identifies where it is.
- A route identifies a way to get there.

The IPv4 address collapses these three notions into an elegant, but sometimes troublesome entity. Problems arise due to interdependencies between the three notions and also because one of the notions cannot be changed without affecting the others.
1.2. PROBLEM AREAS

Mobility

In recent times, mobile devices have been connecting to the Internet in the shape of mobile phones, laptops and PDAs. Typically in the Internet, devices are identified by their IP address. This is a crucial point. By using Shoch’s definitions, as mentioned above, immediately we can see the tension in using an address as a name. If an address changes when a mobile device roams to a new network, it has to all intents and purposes a different name. This becomes a problem for certain applications such a Voice over IP (VoIP) where a mobile user may roam between multiple networks during the course of a conversation. Also certain security applications such as SSH (refx) or IPsec (refx) identify a host by its IP address, a source of much frustration for the mobile user wishing to access a secure resource. Finally it also becomes difficult if a mobile device is running a service which others wish to use (for example, sharing a printer or Internet connection at a conference) since the name of the device is determined whatever IP address the device may currently have. Finding some way of separating identity from location is a key problem area of network architecture research.

Flexible Naming

The Internet at the moment uses DNS to name its endpoints, as well as using IP addresses. DNS names are used as human readable names whereas IP addresses are used as machine readable names. Certain applications such as content delivery networks (CDNs) and load balancing systems have began to code their requirements into the DNS system since there is no other natural way for these applications to satisfy their indirection requirements. However this is an overloading of the DNS system and therefore many resolution steps have to be made in order for the necessary redirection to be performed\(^2\). A URL specifies a host which contains the content that a client wishes to retrieve and this DNS name is bound to an IP address. However it is not always necessary to go to the host specified in the URL to retrieve the content if there may be a proxy cache nearby such as one set up by an ISP or by a dedicated CDN such as Akamai. There is currently no architecturally pure way to do this inside the current Internet infrastructure as the Internet does not host any inherent indirection mechanisms either through naming or through an explicit indirection layer. Ideally we would be able to resolve a URI to a content server which is “nearby” to the client machine. This would enable us to more efficiently use our network resources by not incurring excessive round trips to hosts located in different continents or in networks that the client has poor connectivity to. This is currently not possible since a name is also an address, and a name other than an IPv4 address will simply not be understood by the Internet. If the IPv4 naming and addressing were separated then it would be possible to use arbitrary names and resolves names such as a DNS name or a URI (Uniform Resource Identifier) to an address in a flexible way. This would mean that the difficulties in dealing with mobility, content naming and middleboxes would be greatly reduced. We would therefore have the ability to resolve an identifier to a different entity depending on what functionality we desired. For example, we could resolve an identifier to the “closest”

\(^2\)A brief description of a typical Akamai CDN session can be found here: http://www.cs.washington.edu/homes/ratul/akamai.html
or “best” server depending on our specification.

Middleboxes

The original Internet Architecture assumed a transparent network which sat between various communicating endpoints. Since the original vision was developed, various middleboxes have arisen in the network in the shape of Network Address Translation (NAT) boxes, Proxies (e.g., web caches or content transformation proxies) and Firewalls. These devices interpose themselves between the communicating endpoints (hence the name middleboxes) and can easily cause problems for certain applications. This happens for a number of reasons: firstly, because the network between the communicating endpoints is supposed to be network layer only and middleboxes typically work at the transport and application layers as well. The implications of this are that application-specific code or requirements become embedded inside the network instead of only residing on the end systems. This results in a brittle network since the any change in the layers above the network layer will cause NAT boxes to fail. One example is the difficulty in deploying new transport protocols such as DCCP or SCTP since various middleboxes can only handle TCP, UCP and ICMP packets. Often raw IP packets are also discarded. Secondly, middleboxes can filter out certain packets, or by their very design not let certain packets through. NAT is a classic example of this since it performs site isolation by translating the RFC1918 private IP address ranges to the globally routeable IP address of the NAT box. This means that the issue of incoming traffic to a host behind the NAT i.e., in the network that the NAT is in front of, rapidly becomes very complex. NAT traversal mechanisms are necessarily a hack since they are trying to work around a black box NAT of which is known only a little. There are several different types of NAT (refx), each with their own set of peculiarities which the potential NAT traverser has to work around. Many applications such as web browsing, email and instant messaging work fine through a NAT without any tweaking. However, there are a more complex set of applications which have problems with NAT due to the assumptions that they applications make regarding connectivity. For example, Video conferencing, VoIP, online gaming and peer-to-peer file sharing applications. The problems arise mainly because these applications assume symmetric connectivity i.e., that both communicating endpoints can connect to each other.

1.2.3 Interconnecting multiple types of networks

Since the Internet was conceived, there has been growth in different types of computer networks. The telephony network has been augmented by mobile cellular networks. Wireless LANs, Ad-hoc networks, mesh networks, sensor networks and Personal Area Networks (PANs) have also arisen during this time. Since the Internet has become the network of choice, the TCP/IP protocol suite has been ported to run on these networks. However, it is far from clear that this is the only way to interconnect different types of networks. The Internet model attempts to impose homogeneity on the different types of network which it interconnects. Whilst this approach is elegant and allows for a uniform approach to interacting with different types of networks, it necessarily abstracts away from the details of the underlying networks themselves. This can lead to
difficulties in situations where the structure of the underlying networks are completely
different to the Internet. A typical example of this are sensor networks where the
addressing structures are built around data rather than around nodes. This means that
in order for such a network to be interconnected to the Internet, compromises need
to be made in how accurately the Internet model or architecture can reflect the exact
nature of these networks.

1.3 Thesis Research Areas

The first research topic of this thesis is the separation of network layer naming and
addressing. Currently the network layer collapses both naming and addressing into
one entity, namely the IPv4 address. Although alternative naming exists through the
usage of DNS, IPv4 addresses are still used as identifiers in mobility and middlebox
scenarios. Our approach is as follows:

1. The introduction of a meta-architecture which provides indirection support at the
   lower layer of the software part of the networking stack.

2. Separating naming and addressing so that any arbitrary name can be resolved into
   address inside the context of the meta-architecture

3. Provide a way of hosting various network architectures by the meta-architecture

4. Support the discovery and addressing of these hosted architectures via the meta-
   architecture

We have taken indirection as the main theme of this thesis. We believe that a flexible
indirection primitive at the bottom of the network stack will provide a way to overcome
these problems of deployment and management. In the context of this thesis we have
developed an indirection architecture for the Internet called SelNet. Indirection is a
very powerful tool as it allows us to have late-binding in our system. Late-binding is a
concept borrowed from the field of Software Engineering and describes the process by
which functions can be added to a system at run-time rather than at compile-time. The
problems described above are typically due to the inability to extend the Internet in a
clean way. Clean in this sense means to be extended in a way that does not make the
architecture more brittle by introducing more interdependencies between the layers.
For example, NAT boxes introduce more interdependencies since they rewrite both the
network and transport layer headers. We note here that given the importance of the
Internet as a global communications infrastructure, it is not possible to take a “flag
day” approach (refx) to solving the problems, the Internet must be changed whilst it is
operating.

Late-binding via indirection is one solution to this problem as it allows packets to
be redirected away from the typical processing path and to a function which has been
bound at run-time. Active Networks can be seen as one solution to this problem, but
the trade-off in terms of complexity associated with dynamic code instantiation (espe-
cially in the area of security) turned out to not be worth the benefit of this flexibility.
The balance of this trade-off may change in time, but currently this appears to be the
case. SelNet takes a more restrictive approach, hoping to reduce the complexity side of the trade-off by only allowing indirection to a handful of very specific and well-known functions which may be late-bound. In this sense, it is not so much a network architecture in itself as a meta-architecture which hosts other architecture. This is because SelNet does not mandate what the functions should contain, but rather provides the hosting infrastructure for other network architectures which are instantiated through these functions. One simple example would be the hosting of an IPv4 network architecture on top of SelNet. Indeed, this was the very first experiment that was performed with SelNet. The advantage of an indirection/late-binding approach is that, once the meta-architecture is in place, it becomes relatively straight-forward and clean to extend an architecture which is hosted by the meta-architecture. This is because the meta-architecture can decide which of the host architecture should receive which packets. The process is illustrated in figure 1.2.

![Figure 1.2: Meta-Architecture](image)

The entities of the SelNet meta-architecture i.e., the architectures hosted by SelNet, need to be named and addressed. In SelNet we aim to separate these two concepts by introducing an extensible resolution mechanism which allows a name to be specified and then resolved to an address which corresponds to the appropriate architecture. If we do not have a collapsed name and address entity like the IPv4 address we need to have a mechanism to resolve names to addresses. SelNet has such a mechanism called the eXtensible Resolution Protocol (XRP). XRP is used to resolve from arbitrary names to SelNet addresses. These SelNet addresses are Simple Active Packet Format (SAPF) selectors. These are used by SelNet to demultiplex packets to the correct hosted architecture. XRP and SAPF will be described in detail in the rest of the thesis. Since XRP is used to resolve from names to addresses, it also performs location and discovery services.

### 1.4 Contributions

My scientific contributions are:

- The specification and design of an indirection layer for the Internet which leads to a new network architecture that attempts to reach the goals specified in section 1.3. This architecture allows a very high degree of flexibility whilst maintaining backwards compatibility with the existing IPv4 network architecture.
1.4. CONTRIBUTIONS

• A variety of implementations and testing both in the real-world and in simulation which demonstrate the feasibility of the architecture and its ability to reach its goals. To this end there have been a variety of implementations described in detail in the next section.

1.4.1 Implementation Work

During the course of this thesis, several implementations of SelNet were created which I was directly or indirectly involved with.

**SelNet v0.1**: The very first implementation of SelNet was written by Christian Tschudin and became LUNAR. This version ran only over Ethernet.

**LUNAR**: LUNAR now exists in two main forms: user-space and kernel-space versions. These were also both written by Christian Tschudin.

**microLUNAR**: This was a stripped-down version of LUNAR which could run on the Lego Mindstorms. It used the microIP stack and Van Jacobson header compression. It was written by the Datakom II microLUNAR project students and was supervised by me.

**LUNAR over Bluetooth**: As a follow-on to the microLUNAR project, we ported microLUNAR to run over Bluetooth. The goal of this implementation was to investigate how LUNAR would behave if it was placed in an environment where the link layer is not connection-less (e.g., WaveLAN or infra-red) but connection orientated. Bluetooth requires the setting up of pair-wise connections between nodes before communication. Additionally, instead of using a broadcast packet to discover neighbouring nodes, Bluetooth uses an inquiry phase to build up a neighbour set which an application can then query. This inquiry phase must complete in order for an application to be able to communicate with its neighbours, if it does not know the hardware address of a neighbouring node in advance. Whilst this inquiry is not so costly to do when an application is started, it becomes very costly if it has to be performed regularly to keep track of topology changes in the network. LUNAR over Bluetooth was written by Olof Rensfelt and supervised by me.

**SelNet v1.0**: This version of SelNet included a UDP Tunnelling support for SelNet. This was implemented by myself. The way that SelNet is structured means that implementing SelNet on top of other datagram services other than Ethernet was not too difficult since the principle remains the same. However, tunnelling all SelNet traffic over UDP presented some difficulties since the packets must be received on the virtual interface, be read in from the UDP socket and then de-multiplexed into the SAPF table. This process can be seen if figure (refx). The first SelNet implementations were inserted between the link layer and the network layer, with UDP tunnelling, SelNet functionality was inserted between the transport layer and the application layer. It is our experience that inserting functionality between the layers of the networking stack leads to complications and
brittleness since the principle of layer isolation is not preserved. However, once the cabling between the UDP tunnelling functionality and the Ethernet tunnelling had been established, the network virtualization of SelNet was complete. To the IP stack that was attached to the SelNet virtualized link layer, it made absolutely no difference whether the packets that it was sending out and receiving were been sent out over Ethernet or UDP. One experiment involved sending out packets over UDP and receiving the replies over Ethernet. The IP stack only saw packets departing and arriving as if they were being sent to the link layer. We successfully transferred a 100MB file using this approach and ran an interactive SSH session.

**NS2 SelNet**: In order to better appreciate how SelNet behaves in large networks and to investigate route selection as a SelNet application, I implemented a version of SelNet for the NS2 network simulator. Once again, the difficulty in implementing network software which fits between the layers of the network stack became apparent. The NS2 model of the networking stack is very rigid and not easily extensible since it was not designed to be used in that way. The focus of NS2 is on the transport and application layer, and it is relatively straightforward to extend these layers. Extending the network layer or inserting half-layers is considerably more difficult, requiring the re-wiring of NS2 and subsequently a large amount of layer violations in order to propagate the required information to the desired place.

**MPLS & SelNet**: One of the systems most similar to SelNet is MPLS. In order to further investigate these two systems we internetworked the two systems together so that a single path through the network could be constructed which would be part MPLS and part SelNet. This path would be completely transparent to the applications running over it. This work was done by Andreas Westling as part of a Masters thesis and supervised by me.

**Janus**: In order to investigate interconnecting different types of networks together, we applied the SelNet principles to interconnecting a IP network with a sensor network. An implementation was created which allowed us to send SelNet queries into the sensor network and report them to a host running a TCP/IP stack. Additionally, the sensor network was able to react to an event and send a message to the host outside of the sensor network. An additional implementation iteration allowed us to take this approach one step further and actually inject code into the sensor network. This implementation was written by Mats Uddenfeldt and Björn Ahlström and supervised by me.

We considered for a while the prospect of porting SelNet to the Planet Lab, however this approach was quickly discarded since currently it is not possible to have full root access on a Planet Lab slice, so kernel modules can not be loaded and network virtualization tools such as TUN/TAP (used in the userspace versions of SelNet and LUNAR) also can not be used. This will hopefully be changed in some future version of Planet Lab when more virtualization infrastructure can be deployed.
Chapter 2

Related Work

2.1 Introduction

Related work can be divided into a number of different areas:

1. Underlay Networks
2. Network Architectures
3. Overlay Networks

2.2 Underlay Networks

2.2.1 MPLS

Multi Protocol Label Switching (MPLS) is an underlay network which uses label switching in the network for faster and simpler packet forwarding. The motivation behind this is to provide functionality to IP networks that currently do not exist. For example, traffic engineering, Quality of Service (QoS) and Virtual Private Networks (VPNs). MPLS achieves this functionality by operating beneath the IP stack. By positioning itself there, MPLS is able to make decisions about how to treat traffic before the traffic reaches the IP stack. MPLS adds a label to a packet when it enters an MPLS enabled network. This label identifies a Forwarding Equivalence Class (FEC) which tells MPLS how to forward the labeled packet. All packets belonging to the same FEC are treated in the same way. This allows MPLS to perform QoS amongst other things as it thereby has control over a particular flow of packets. When a 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.

This approach is very similar to SelNet in that we also build our own network underneath IP. We also provide a number of different functions which can be used to treat
packets in different ways. Additionally, MPLS and SelNet both use label switching underneath IP for packet forwarding. The biggest difference between SelNet and MPLS lies in how the control plane, rather than the forwarding plane, works. In SelNet, the end-system expresses its needs by using the XRP protocol to steer the resolution process in the network. MPLS does not provide such a mechanism to the end system as it was constructed as a tool to assist providers in controlling their networks. The different goals of the two projects clearly delineate their differences. Also, MPLS specifies quite clearly which functions should be present in the network. SelNet does not specify which functions should be present, but how any arbitrary function can be accessed.

2.3 Network Architectures

2.3.1 Bananas

Bananas is a project to enable explicit multi-path routing in the Internet. The motivation behind the project is to examine the value of multi-path routing and to create an evolutionary framework to enable it in the Internet. Bananas proposes the usage of a PathID to address the multiple paths available in the network. The PathID is typically a hash of the links that need to be traversed on the path. A hash provides for a more compact representation than a concatenation of IPv4 or IPv6 addresses. A reversible hash can be used in order for intermediate routers to examine the full path. In order for the framework to permit incremental deployment, two things must be present. Firstly, the framework must interact with the Internet’s routing architecture and secondly, it must be able to cope with the fact that portions of the Internet will not support its extensions. In order to cope with the first issue, the authors propose a number of extensions to existing routing protocols such as BGP and OSPF which will allow them to inter-operate with Bananas. The authors propose a path elimination algorithm which allows them to deal with the second problem. This algorithm computes which paths are available to Bananas.

SelNet also attempts to open up the myriad of paths in the Internet to the end-system, but the approach taken is quite different. The approach taken by Bananas is elegant and well-thought through. Bananas is also a very focussed project, working on a very specific problem. The difference in focus is the largest contrast between SelNet and Bananas and is what motivates the different design decisions between the two projects. SelNet attempts to provide the maximum amount of flexibility in how the network can behave. Bananas attempts to provide the most efficient mechanism to allow multiple paths to be used in the Internet. The choice of the PathID identifier and how this is constructed clearly reflects the goals of the project. SelNet’s choice of an opaque selector is chosen for maximum flexibility as the selector itself could refer to any function. This means that it is not as efficient as the Bananas PathID for multipath routing. However, it is more flexible than then PathID as it has no inherent structure so there are fewer restrictions on how it can be interpreted.
2.3. NETWORK ARCHITECTURES

2.3.2 TRIAD

TRIAD is an explicit content layer for the Internet. The rationale behind this project is that since so much of the Internet’s traffic is based on locating and transferring content it makes sense to add additional mechanisms to the Internet architecture to support this. TRIAD provides a resolution system which provides content-to-name or content-to-number resolution. They define an explicit content layer above IP and use content-aware routers and DNS servers to introduce their new resolution functionality into the network. When a client attempts to access a URL, the server name encoded in this URL is lookup up in the TRIAD-extended DNS system. This lookup is then forwarded through TRIAD until it reaches the “best” content server for this particular client. The definition of best can vary from client to client. The content routers in TRIAD exchange content reachability with each other in order to populate the routing tables by using the Named Based Routing Protocol (NBRP). NBRP is similar to BGP in the way that it is distributes reachability information across domains.

TRIAD points the way to how content-based routing could be introduced into the Internet. It emphasizes trying to be backwards compatible as much as possible and advocates changing NATs, BGP routers and DNS instead of attempting to replace the entire Internet infrastructure. We have also thought about content-based routing in SelNet and it seems to us that TRIAD and SelNet are complementary in their approaches. TRIAD deals with locating content in the network. SelNet deals with mechanisms which can support the location of content and the subsequent packet forwarding mechanisms. It would be possible to express queries for content using XRP, then using TRIAD to locate the appropriate content server through its protocols and then forward packets to the server using SAPF. Indeed, projects such as TRIAD inspired us to work on the original SelNet proposal to see if we could create a network architecture which would provide explicit support for alternative styles of routing such as content-based routing.

2.3.3 Role Based Architecture

The Role Based Architecture (RBA) is a proposal to investigate how a network architecture could look which does not have a layered model like the TCP/IP or OSI models. Whilst the layered model provides structure and standardization to the Internet architecture, it encounters problems with as attempts are made to evolve the Internet. This is because there is no natural way to extend the functionality of the Internet. As a result, new functionality is typically inserted between the layers. For example, TLS at layer 4.5, IPSec at layer 3.5 and MPLS at layer 2.5. Simultaneously, middleboxes have become increasingly popular in the Internet for proxy, firewall and NAT functionality. Since these middleboxes typically look across multiple layers, they cope very poorly with new functionality which is introduced between the standard layers. The RBA proposal seeks to cope with these problems by eliminating layers entirely i.e., instead of a protocol stack they propose a protocol heap. In the RBA model, the various protocol actions (Roles) are decomposed into functions which are addressed by a heap of headers carried in a packet. Since the order of protocol actions is determined by the packet, this model explicitly allows for middleboxes. This is because in order to add a new
protocol action, it is sufficient to simply add another protocol header to a packet. Since
the protocol actions on a host or router are applied in the order specified in the packet,
adding a new protocol action does not violate any existing architectural assumptions.

SelNet and RBA share similar goals in wishing to extend the functionality of the
Internet. SelNet does not specify how the layering of the network architecture should
be implemented. In fact, it would be possible to implement a version of RBA using
SelNet. A packet could carry selectors which reference various protocol actions which
reside on a host. In order to extend the functionality of the network layer, it would be
sufficient to add an additional selector to the packet header so that a new protocol action
can be invoked. We see RBA more as a architectural viewpoint than as a competitor to
SelNet. The approach or vision of the RBA proposal has informed our own work on
SelNet.

2.3.4 Metanet

Metanet is a proposal which is an alternative to the Internet architecture. It suggests
that instead of attempting to impose a single homogenous structure on the Internet,
regions should be constructed instead which shared some common attributes. The mo-
tivation behind this is that it allows benefits in terms of efficiency and deployability.
The complexity involved of imposing a single homogenous structure is moved to inter-
facing between different regions in the Metanet. This allows for local optimizations to
be made which do not need to interfere with the operation of the rest of the Metanet.
As long as a region conforms to an agreed interface with its neighbouring regions, it
is free to change its internal representation. This allows for modularity. Additionally
this architecture embraces middleboxes such as NATs, proxies and firewalls since they
define the boundaries of a region.

The SelNet approach can be seen as an attempt to build a Metanet. XRP can be
used as the protocol to traverse regions. This is what we have done as described in
sections [3.4] and [3.5]. Our research contribution has been to solve the technical issues
with taking the approach delineated in the Metanet white paper. We note here the
foresight of the authors of suggesting such an approach in 1997 which has inspired a
number of subsequent proposals.

2.3.5 Yellow Book

Yellow Book transport services (or just Yellow Book) was a proposed standard for a
transport service which enabled communication between disparate networks. It was
developed at University College London in 1980. Yellow Book concatenated different
address types together and then used gateways between different types of networks
to translate between the different types of addresses. It was developed to deal with
interconnecting between networks such as X.121 and TCP/IP.

One of the differences between Yellow Book and SelNet is that SelNet attempts to
position itself under protocols such as TCP or X.121 and thereby find a common point
for uniform addressing (i.e., SAPF selectors). This by-passes the upper layers of the
protocol stack whereas with Yellow Book, the transport level addresses are preserved
and concatenated together to perform source routing through the network.
MORE – design trade-off discussion

2.3.6 Active Names

Active Names [2] 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.

2.3.7 IPNL

Resolution and translation mechanisms have also been investigated by IPNL [?]. In IPNL (IP Next Level), the authors propose a new Internet architecture based upon an extended version of NAT. The motivation behind this approach is to provide the benefits of NAT simultaneously with the benefits of IPv4. The authors argue that the reason that NAT is deployed is not only to deal with the address space exhaustion issue, but also because NAT provides site isolation. This has numerous administrative and security-based advantages. The disadvantage of NAT is that it breaks the symmetric connectivity principal of the Internet because some hosts may reside in domains which cannot be directly addressed from the Internet. IPNL therefore attempts to reconcile NAT and IPv4 together in a cohesive architecture. In order to achieve this the authors use Fully Qualified Domain Names as global end-to-end addresses and then use IPNL addresses to perform dynamic translation between different domains. This approach is intriguing as it advocates making routing decisions based on a name rather than an address as in the current Internet routing architecture. Also, by creating a layer above IPv4 i.e., essentially using IPv4 as a link layer, the authors ensure that only NAT boxes and end-systems need to be upgraded in order for IPNL to be useful.

SelNet also attempts to provide different domains in the network and offer some translation mechanisms in order to map between them. However, IPNL is much more focussed on solving the specific problem of how to provide both the benefits of NAT and IPv4. It makes no claims to providing in-network processing or redirection to specific middleboxes, for example. Additionally, instead of tightly integrating our system with IP in the way that IPNL functions, we place our translation mechanisms below IP in order to avoid dealing with the complexity of IP operations. The trade-off being that IPNL is potentially more well-suited than SelNet as a near-term solution, but perhaps SelNet is more suitable for a longer-term solution which could potentially replace IPv4 as it is more flexible.
2.3.8 Nimrod

Nimrod (refx) defines itself as a “A new IP routing and addressing architecture”. It is notable as being one of the first projects to concentrate on the architectural issues of the Internet rather than the implementation issues of one or more layers of the existing Internet. Two of the main design goals of Nimrod were scalability and policy-based routing. The beliefs of the project were that if these goals were met, then the architecture would be sufficiently powerful and flexible. Nimrod is centered around a link-state approach for routing which separates route calculation and information distribution. The addressing mechanism in Nimrod would be hierarchical in order to allow information hiding and abstraction. This is an important part of the scalability approach of Nimrod, in a similar way to how BGP provides scalability through hierarchy. Another key point of the Nimrod design space is the separation of identity and location in order to cope with some of the issues of the current Internet such as mobility. Despite that Nimrod provides an efficient datagram forwarding service, the fundamental mechanism for forwarding in terms of an abstract concept is “flow”. This provides a useful mechanism for scaling analyses of packet flows through the network.

Nimrod is a more ambitious project than SelNet as it attempts to cover all aspects of a network architecture. Nimrod proposes a complete naming, addressing, forwarding and routing architecture for the Internet. From the the routing algorithms that should be used to the packet format for forwarding, the Nimrod proposal has a great deal of depth and insight as to how these could be designed and implemented. SelNet, however, provides the ability for different network architectures to be implemented on top of it, but does not specify how this should be done. Additionally, Nimrod does not present an architecture for indirection, it does not address how different types of functionality could be addressed inside of the network. Nimrod represents one of the first detailed proposals and designs for a new network architecture. The scope and vision of Nimrod remains impressive almost a decade later.

2.3.9 Plutarch

Plutarch [3] 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. It is worth noting here the link that both Plutarch and SelNet share with the Metanet project in terms of proposing homogeneous contexts and translation mechanisms between them.
2.4 Overlay Networks

2.4.1 Internet Indirection Infrastructure

The fact that the Internet suffers from being overly direct has also been observed by the Internet Indirection Infrastructure (i3) project \[4\]. 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. By allowing hosts to communicate with each other indirectly, i3 enables effective multicast and anycast support. This is because in these abstractions, the sending host does not know the identities of the receiving hosts. Additionally, the i3 approach provides host mobility support because the indirection hides the location of the receiving host.

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.

2.4.2 Layered Naming Architecture

The Layered Naming Architecture position paper advocates adding two additional naming layers to the current Internet architecture. One layer names services in the networks with service identifiers (SIDs) and the other layer names end points in the network with EIDs end host identifiers. The purpose of these two additional layers is to correct a number of issues with the current Internet architecture and by that providing explicit architectural support for mobility, middleboxes and modest support against Distributed Denial of Service (DDoS) attacks. The proposal first tries to correct the flaws in the current Internet naming systems and then demonstrate how a corrected system could enable an architecturally pleasing set of applications previously not possible with the current Internet architecture. The introduction of these two new naming layers allow services and data to become “first class” objects of the Internet. Ensuring a separation between an identifier that represents a service and an identifier that represents an endpoint allows mobility to be easily introduced into the Internet. This is because the collapsing of identity, access and location into the IPv4 address is no longer present. The Layered Naming Architecture proposal also introduces the notions of intermediaries and delegation as replacements for middleboxes in the current Internet architecture. An endpoint can delegate packet receiving capabilities to an intermediary somewhere else in the network by inserting a rule into the resolution infrastructure. This is how proxies, NATs and Firewalls can be cleanly re-introduced into the Internet. SelNet shares similar goals to this proposal, but is comparable to only a certain part of the proposal. SelNet resolves SIDs to selectors which are then used for forwarding, thus avoiding the EIDs and IP routing/forwarding of the Layered Naming Architecture.
2.4.3 Resilient Overlay Networks

The Resilient Overlay Networks (RON) project [5] 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 plus the topology and capacity mismatch between the overlay and the underlying network. We advocate an underlay approach which strikes a balance between the needs of the users and the network operators will be a more appropriate long term solution.
Chapter 3

Summaries of the Papers

3.1 Paper A: Network Pointers

The first published paper laid the architectural foundation for the SelNet architecture. It investigated an alternative way of structuring a network based around the concept of Network Pointers. This concept was born as a result of the notion that the Internet is overly direct. This notion arose from observing the growth of overlay networks such as Peer-to-Peer Networks, Virtual Private Networks (VPNs) and route optimization overlays (such as RON refx). The commonality between these different approaches is that they attempt to provide indirection by building a new network over the existing infrastructure. By redirecting packets into a virtualized overlay network the overlay itself can choose how to route packets. This is a marked difference from being completely dependent on the routing configuration of the networking infrastructure below the application. The main reason for wishing to build a new infrastructure on top of the existing one is that there is some functionality which is desired but the existing infrastructure does not provide. For example, customized routing decisions or media transcoding. We generalize this to say that if indirection was present at the core of the network architecture, then it would not be necessary to implement overlays. The application, instead of building an overlay, would simply request the network to redirect its packets to a certain function which would be controlled by the application. Network Pointers are one way of achieving this indirection. In summary a network pointer consists of a pointer value and a packet processing function in order to provide indirection support. Unlike other indirection proposals, we do not use IPv4 addresses as part of our new architecture but rather create a separate namespace under IP which consists of local names which can be dynamically resolved. This allows us to disentangle identity, access and location which are typically conflated in the IPv4 address.

Insert diagram

The contribution of this paper is the establishment of the concept of a Network Pointer. This concept is then used in three different scenarios in the paper to show how they can benefit from the use of Network Pointers. We then describe in more detail how Network Pointers can be constructed, how they can be resolved and addressed.
Implementation details are then discussed. This describes the first implementation of SelNet and its functionality.

My contribution to this paper was as co-author. I invented and investigated with some preliminary implementation work the asymmetric connectivity scenario. I participated in many discussions with Christian Tschudin concerning all of the scenarios and propositions made in the paper.

3.2 Paper B: SelNet: A Virtualized Link Layer

Following on from the work detailed in section 3.1 we present SelNet: A Virtualized Link Layer which is one possible instantiation of the network pointers system. It provides a indirection layer which can host the existing Internet as well as additional infrastructures. We present in this paper a complete network architecture based on the network pointer idea which is positioned as a virtual link layer underneath the existing IPv4 network. We show how forwarding via selectors can be performed and how dynamic resolution of names is achieved.

SelNet is a network architecture which is positioned in the current context of the Internet architecture as a virtualized link layer. It provides explicitly indirect support to the upper layers of the protocol stack which can be used for not only network layer indirection such as loose source routing but also application layer indirection mechanisms such as distributed proxy networks. 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, SelNet aims to achieve this by the usage of explicit resolution mechanisms and addressing packet processing functions rather than nodes.

The contribution of this paper was the establishment of SelNet as an implementation of the Network Pointers concept. It describes in detail how SelNet actually functions and goes through, step-by-step, a number of scenarios. We show how static forwarding is accomplished in SelNet by the demultiplexing of SAPF selectors and how these selectors can be dynamically allocated by the usage of the XRP protocol. With these SelNet primitives, we illustrate how SelNet can be used to implement NAT-style site isolation scenarios without the brittleness of a typical NAT solution. We also show how route or path selection can be achieved with SelNet and how this allows SelNet to scale. We close with some discussion of SelNet mechanisms for soft-state and protection from Distributed Denial of Service (DDoS) attacks.

I was the lead author of the paper and wrote most of the text. I had frequent discussions with Per Gunningberg which were of enormous help in clarifying my thoughts. Christian Tschudin helped me with discussions and some vital re-drafting of the paper. Ian Wakeman also contributed general feedback on the paper.

1 "The Selector Network": so named as it uses selectors at the core of its design.
3.3 Paper C: LUNAR: Lightweight Underlay Network Ad-hoc Routing Protocol and Implementation

The first implementation of SelNet was the LUNAR ad-hoc routing protocol. It is targeted at small ad-hoc networks, approximately a dozen nodes with a maximum radius of no more than three hops. The aim with LUNAR was to explore novel ad hoc routing strategies and to constrain ad hoc routing protocol design to pragmatic boundaries. Low protocol complexity helps to easily implement LUNAR in other environments, as we demonstrate with $\mu$LUNAR and also form a good starting point for adding the important self-configuration elements. The main difference between LUNAR and existing ad-hoc routing protocols like AODV, DSR and OLSR is that it is based on SelNet. LUNAR therefore positions itself underneath IP and uses the XRP mechanisms for dynamic resolution and SAPF for packet forwarding. LUNAR targets the common-case of network clouds with approximately a dozen nodes and a diameter of up to three hops (refx). We believe that such settings will be the most popular ones where ad hoc networks can and will be put into operation.

This paper contributes the first application of SelNet, namely an Ad-hoc routing protocol. It describes how SelNet can be used as a foundation for such a protocol. We show the components of SelNet and how they can be used to build on-demand forwarding paths suitable for an Ad-hoc environment. In particular the XRP message formats used for discovering available routes. The various implementations of LUNAR are presented: LUNAR, $\mu$LUNAR for embedded systems, Bluetooth LUNAR and a port to Windows XP/2000. Evaluation results are subsequently presented and discussed which show LUNAR to have equivalent or better performance than existing Ad-hoc routing protocols. We also present some preliminary work on formal methods which involves the verification of the LUNAR protocol in the SPIN model checker.

I was a co-author of this paper and contributed some of the text. I supervised the implementation of microLUNAR (which was a student project) and Bluetooth LUNAR (which was a Masters thesis project). I was also the co-designed of LUNAR together with Christian Tschudin and performed many tests and evaluations of it.

3.4 Paper D: Janus An Architecture for Flexible Access to Sensor Networks

In order to investigate how SelNet could be used for interconnecting different types of networks, we attempted to interconnect an Internet-style network with a sensor network. The motivation behind this was to provide a flexible way to access a sensor network. This is desirable as sensor networks are typically deployed in harsh or remote environments where it is hard to change or upgrade the nodes of the network. We designed and built an architecture called Janus which is based on SelNet. Janus was used to interconnect between a host in an IP network and a 14 node sensor network. Typically it is challenging to interconnect two such disparate networks since they operate on fundamentally different principles. The clearest illustration of this is the addressing

$^2$Lightweight Underlay Network Ad-hoc Routing Protocol
structure of the two networks. The addressable component of an IP network is the IP
address, whereas in a sensor network it is often the case that the addressable compo-
nent is a piece of data or a function. These two types of addressing can be contrasted
as node-centric and data-centric. We feel that a SelNet-style approach is useful in this
context since SelNet uses functions at its heart which maps very well to the data-centric
model of sensor networks. Additionally, we take a lightweight approach where primitive
functions on a gateway to the sensor network can be combined at an external host
to provide service composition. This is in contrast to existing RPC approaches where
the server contains comparitively more functionality than the client. We believe that
it is critical for as little functionality as possible to be present in the gateway. This is
because the gateway will usually be in a location which is hard to access physically and
additionally since the gateway may be a sensor network node itself. We present in the
paper a description of the Janus architecture, show how it is implemented and discuss
some of the issues that arise with our architecture.

I was one of the main authors of this paper, I supervised the SelNet implementation
work and wrote the majority of the text.

3.5 Paper E: Janus: An Active Middleware for Access-
ing Sensor Networks

The previous paper described the initial design and implementation of Janus. This
paper describes Janus more fully and also presents the code deployment mechanisms
that we have developed for Janus. Our motivation for devising a code deployment
mechanism for sensor networks is the logical extension of the rationale presented in
section 3.4. Providing service composition for the sensor network is useful when the
services available can be combined to meet the requirements. We believe that it will be
useful for sensor network deployments if the nodes in the sensor network can be dy-
amically reprogrammed. This allows for the sensor network to be responsive to future
application requirements. The main focus of this paper is to show how code deploy-
ment fits naturally into the Janus framework. We have ported the µLUNAR system to
the Contiki operating system and used it for the basis of our code deployment platform.
The reason behind this is that we need to address particular nodes in the network in or-
der to reprogram them, and µLUNAR gives us the ability to communicate with any
node in the sensor network by giving us ad-hoc, multi-hop routing. We then extended
µLUNAR to carry code in the XRP messages. This code could then be inserted into
the SAPF table inside a node in the sensor network. Once this code had been inserted
it could be addressed in the same way as existing functions in the sensor network. The
paper describes how we have implemented this functionality. We note here that it took
one person, who was unfamiliar with the codebase, only two months part-time to port
µLUNAR to Contiki and add loadable module support.

I was the main author of this paper, supervised the implementation work and wrote
nearly all of the text.
Chapter 4

Conclusions and Future Work

This thesis has mainly been concerned with the description of an indirection architecture, in order to solve the problems that are present with this existing architecture. Despite its phenomonenal success, the Internet of today has many problems, mainly due to it becoming a victim of its own success and thus being used in ways that its original creators never intended. Amongst the imminent problems that the Internet currently faces are: address space exhaustion, routing, security and general “Internet Artieosclerosis” (refx!). These are all areas that border on the alternative architecture presented in this thesis, but it is the area of indirection that is the main focus. SelNet (or the “Selector Network”) is a new network architecture which attempts to bring indirection to the bottom of the networking stack by the creation of an explicit indirection layer underneath the existing Internet. Indirection in this context means the ability to control, with relative ease, the paths that packets take through the network. In the current Internet, at least for the access networks which comprise the majority of users, there is one and only way that a packet can go and that is to the default route. Whilst this is sufficient for many applications, numerous applications have arisen which require indirection to be present not only at the application-layer of the network, but at the network-layer itself. For example, the usage of proxies or middleboxes requires the positioning of a network entity between a pair of communicating endpoints. This is not so difficult if the proxy lies already on the path between these two endpoints, but it rapidly becomes a difficult problem if it does not. Another example is the selection of routes or paths through the network. Currently the user has no influence on the path that his packets will take through the network. This means that if the primary path between a pair of communicating endpoints becomes overloaded, a secondary path which may have marginally less capacity than the primary will not be available. This is a phenomenon that has been documented by the Resilient Overlay Networks project (refx!) and leads to both over- and under-utilization of network resources. The difficulty arise because the routers in the core of the Internet are not well-placed to see the big picture of end-to-end paths through the entire network. The endpoints of the Internet are, however, much more well-suited to seeing the quality of the paths between themselves. This has been noted by (refx: simon leinen and refx: the case for separating routers from routing).
SelNet is not the first project to discover that indirection as a primitive is useful for such scenarios as outlined above. The research area that grew around the advent of overlay networks (refx!) is one such area. However, as has been noted elsewhere, the Internet itself evolved as an overlay over the existing telecommunications infrastructure, just going to show that there is nothing new under the sun. The main motivation behind these overlay networks is to build a completely new network over the existing networking infrastructure. This approach is very appealing since it allows the rapid deployment of new networking systems which has traditionally been a real stumbling block to the advancement of networking research. There are drawbacks to the overlay networks such as inefficient routing, inflated headers and lack of reusability i.e., once overlay solution cannot be used for another since the situation would rapidly arise with multiple overlays stacked on top of each other.

SelNet arose from work with Active Networking (refx) and Overlay Networks (esp. Peer-to-Peer). It was our opinion that the overlay approach, that is to say building systems on top of already existing systems, would be a feature of any future Internet architecture. This lead us to the approach of constructing a network architecture that would support overlays as a “first-class object” of the architecture. Since overlays are one mechanism of providing indirection, this links very well to our intuition that indirection is a very powerful primitive to have at the core of the network architecture. The creation of an indirection layer underneath the current Internet is one way of achieving this.

There are two main points to the design philosophy of SelNet. One is that we address functions as the elements of the network architecture rather than nodes or interfaces as in IPv4/v6 or Ethernet. The main reason for this is to maximize the life-time of the architecture. As pointed out in the Invariants paper (refx), the main invariant of the current IPv4 network architecture is the IPv4 address itself. The structure and semantics that are applied to these 32 bits by the various routers and end systems of the Internet are what prevent it from being changed. We believe that nodes or interfaces are already too specific for a network architecture since they limit quite strongly how these addresses can be interpreted in the future. We believe that functions offers enough generality for long-term usage of the architecture. The second main point of the SelNet design philosophy is that we believe that the control of the network architecture should be as much in the hands of the end users as possible. This is mainly for the reasons framed in the classic paper “End-to-end arguments in System Design” (refx) which do not need to be rehashed here.

These three main components: 1. indirection, 2. functions and 3. user-control combine together to provide the basis of the SelNet design. The goals of the SelNet design are to provide a network architecture that is as flexible and as expressive as possible. It should also still be efficient enough for regular usage. It should not be restricted by the network/application layer that we sit on top of, as we would with the overlay case.
4.1 Conclusions

From our work with SelNet we can conclude that there is a lot of value in researching network architectures, even if they are not deployed immediately or even at all. It provides a framework for structuring discourse on how networks should be constructed. It is useful to evaluate and consider alternative approaches to existing systems since it helps us understand better how future systems could be structured.

Through the work performed on route selection we feel that it is possible for SelNet to scale up to large networks since the resolution process does not necessarily have to travel through the entire network. Additionally our work on microLUNAR has shown that even small devices can still support a SelNet-style routing and forwarding system.

4.2 Future Work

The future of SelNet lies in two different domains. Firstly, examining the implications of SelNet and what can be done with SelNet. Secondly the implementation and deployment aspects of SelNet. In the first domain it would be extremely interesting to see what applications does SelNet enable? For example, having the explicit resolution process before actual data transfer can take place means that service discovery can be integrated directly into SelNet. Recent work has been done in applying techniques from Natural Language Processing (NLP) to service discovery and location (refx). It would be potentially useful to apply these techniques to SelNet to provide better service location for the end user. Another aspect of the implications of SelNet would be the economic aspects of SelNet. Which business models are enabled or threatened by SelNet? Would Service Level Agreements (SLAs) between operators be viewed in a different light if user-controlled route selection was enabled?

The second domain concerns implementation and deployment. Currently there is still lacking a solid implementation of SelNet which permits both the route selection and distributed proxy scenarios to be deployed in a real-world setting. Ideally such an implementation would be in the Operating System kernel in order to minimize the performance aspect of an indirection layer being in userspace. The advantage of having such an implementation would mean that we could deploy it to interested parties and examine the experiences of other users using SelNet in slightly more diverse settings than we have currently been able to do. Another interesting possibility would be to port a stable implementation of SelNet to small devices such as the Linksys home router / access point which runs Linux. This would mean that for certain applications we could use a simple hardward platform for deployment in a similar way to Least Cost Routing boxes in the telephony world. Finally, creating a version of SelNet which would be suitable for deployment on the Planet Lab would be another valueable tool to help with the evaluation of SelNet in quasi-realistic situations. Being able to take advantage of the Planet Lab environment would be a first step to seeing how a real deployment of SelNet could actually look like.
Below is placeholder text for mapping SelNet selectors to physical layer wavelengths or frequencies

Trends in networking are optical physical layers for the core and wireless links for the edges. Also demands for sophistication of interaction with the network are increasing. We believe that there is a natural synergy between these trends. We present a new network architecture called Octopus. So named as it is a combination of layers $7 + 1$. The main premise behind Octopus is that it is possible to map an application-layer frame to a fibre wavelength or a radio channel. When a host wishes to communicate with a service or host, it sends a search request to its default gateway. In the IP model, the default gateway is an IP router. In the Octopus model the default gateway is a search engine which has knowledge of many different types of services. Unless a host has already cached the required information to reach a particular host or service, the first action it must take is to perform a search. How this search is performed and how the search terms are specified are implementation details. If the search has been successful, then the gateway returns the information to the requesting host of how to reach its destination host or service. This information contains the physical layer information which should be used to reach the requested host or service. In some situations it may be the case that what is returned is the physical layer information which can be used to reach another gateway which perhaps has more information about the available services in the network.
Bibliography


