A Content Adaptation Network using SelNet

Mats Uddenfeldt and Richard Gold
Department of Information Technology
Uppsala University, Box 337
S-75105 Uppsala, Sweden
Mats.Uddenfeldt.9771@student.uu.se
Richard.Gold@it.uu.se

ABSTRACT

The Internet today shows a constantly increasing number of hosts with more and more disparate capabilities. This leads to a growing need of content adaptation services, which are able to respond to different client capabilities. In order to make such services relatively straight forward to deploy, this paper suggests a framework which uses SelNet, a network architecture that is based on a virtualized link layer with explicit indirection support, as the infrastructure for the content adaptation network. We discuss the implications of such a framework and report on implementation progress.

1. INTRODUCTION

The number of nodes connected to the Internet is continuously increasing and with the introduction of the next generation of mobile systems, this development is expected to continue. At the same time the clients are growing more and more disparate with regard to their different hardware capabilities. Today a content provider is expected to make specific allowances to make sure the information can be presented in a meaningful matter without exceeding the bounds of the capabilities of the connected terminal, which could be a 3G phone, a laptop on 802.11 or a desktop on a modem. The bounds could be described in terms of the size of the display, limiting the size of images, the memory available for storing material and the CPU power or the connection bandwidth. Coupled with this is the increasing demand to be able to use multimedia applications regardless of terminal and network connection.

One solution would be to create a service which is able to respond to the different client capabilities by reshaping the data being relayed back to the terminal. The mobile aware server architecture (MARCH) is one solution to this problem [1]. MARCH is a distributed content adaptation architecture which adapts multimedia content in client-server environments, and the transmission of this content, to match the network connection and terminal capabilities of the connected client. This is done by using a client entity which informs a special server of its capabilities. This mobile aware server (MAS) will then decide which of the possible proxies to use for this session.

This paper will discuss the advantages of implementing a distributed proxy scenario using SelNet as a virtualized link-layer [3], as the infrastructure for the interaction within the content adaptation architecture. SelNet provides a way to build selectors, or function mappings, within the network nodes, which can be used to create a session specific proxy path through the network. This makes it possible to create proxy chains which can process and forward content back to the client independent of further interaction with the configuring server.

To appear on Swedish National Computer Networking Workshop (SNCNW) 2004, Nov.23-24, Karlstad University, Sweden

2. RELATED WORK

Multi Protocol Label Switching (MPLS) is an underlay network which uses label switching into the network for faster and simpler packet forwarding. When a packet enters an MPLS network a label is added to it. This label identifies an action for the next hop. When the packet reaches the boundary of the MPLS enabled network the label is stripped away and regular routing is performed. SelNet, like MPLS, also introduces label switching between the network and link layer. An attempt to internetwork SelNet and MPLS to study their behavior have been made in [5]. MPLS distributes its labels among all MPLS routers, whereas in SelNet the labels are local to each SelNet node. MPLS labels are used to address paths, whereas SelNet labels address functions. The last property gives us extended flexibility as to which functionality we can add to the network.

Plutarch [2] is a network architecture proposal for bridging disjunct networking contexts to form a cohesive network. Contexts are bridged together using interstitial functions (IFs) and provides indirection by the ability of choosing which context to map a particular packet flow on to. The approach of making the heterogeneity in the Internet explicit and controllable is shared by Plutarch and SelNet. Plutarch does not specify mechanisms to perform this task, but leaves it to the actual implementation details of each particular context.

Policy-based content adaptation have been discussed in the mobile aware server architecture (MARCH) [1]. The idea of a centralized point, which depending on the operating conditions of the client terminal device, offer content adaptation in the form of a proxy path is also the foundation of our proposed framework. But instead of implementing this negotiation at the application level, we use the extended functionality of SelNet to implement it at the virtualized link layer. The problems with source routing and making sure the reverse path is handled in the right order can be solved with relative ease thanks to the inherent properties of SelNet.

3. SELNET AND SELECTORS

SelNet uses XRP (eXtensible Resolution Protocol) as a standard means of signaling and SAPF (Simple Active Packet Format) to provide basic demultiplexing functionality [3]. This functionality

is provided through the use of *selectors*, which are function names, giving the name "Selector Network" to SelNet.

For signaling to work within SelNet we must know the ethernet address of the node and the name of a selector to communicate with. XRP is used for resolution in the following way. An XRP resolution request (RREQ) is a broadcast of a name to a well-known selector address on the network. This request specifies the address we wish to resolve and how the resolution should be done. This request propagates until it reaches the target, which will reply using an XRP resolution reply (RREP) with its ethernet address. SAPF defines the the data packet format for SelNet. XRP and SAPF will be mentioned within this paper, but are explained in greater detail in [4].

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 itself. In the following example we assume that we map the virtual link layer to a real ethernet thus reconstructing the current Internet model.

To introduce the concept of how SelNet functions we are going 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 carries the selector as an address field. This packet format is defined by SAPF. The selector address, a 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.

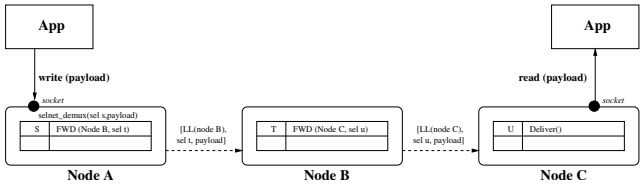

Figure 1: Forwarding in a static SelNet setup. (FWD= forwarding function, LL=link layer)

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 (FWD) is invoked since the selector s 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. selector u on Node C, it 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.

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 the IP traffic will be carried over SelNet. This functionality was implemented in [4].

3.2 Content Adaptation in SelNet

In the previous example we showed how SelNet works when selectors are statically assigned. Static, global selectors function only when all nodes independently agree on the assignment. This is because there is no distribution mechanism, for distributing labels, in SelNet [3]. However, selectors can also be dynamically assigned by communicating nodes since selectors only have validity on the node that assigns them. The selectors at each node are bound to a unique session and can be used to set up adjustable indirection through the network. These indirection properties allow scenarios, such as distributed proxies, to be built relatively easily compared to implementing similar functionality purely at the application layer. It also gives us a single point of indirection with no need of overlapping indirection mechanisms. To introduce a framework of content adaptation in SelNet we will use the following types of selectors: *forward* functions, *proxy* functions, *transcode* functions and *deliver* functions. The forwarding function, as previously explained, will carry a packet to its destination, the proxy function will help set up a proxy path from the client to the server. The transcode function will perform the content adaptation and pass the transcoded data along the path on the way back to the client. Finally the deliver function will pass the packet payload through a socket to a waiting application.

4. FRAMEWORK

Setting up a multi-hop content adaptation path over a chain of distributed proxy servers at the application layer can be very complicated. If there exists an architecture which can handle the adaptation, there are still three questions to be answered: How do we know if the content provider is using a content adaptation system? How do we inform the server entity of our capabilities? How should an adapation path be constructed and imprinted the network? The first two questions are not discussed in this paper, but some methods are explored in [1]. Instead this paper focuses on methods to let the content provider shape the returning traffic with the use of an intermediary proxy provider. We are going to consider a framework containing a client entity, a proxy provider controlling a number of proxies and a server entity, all of which are SelNet nodes.

The task of setting up a session over the adapation network can be split into the following sequence. First the client entity will intercept an application request for an URL and relay it, combined with the user preferences and terminal capabilities, to the proxy provider. The proxy provider will then build a suitable chain of transcoding proxies and set up a path of selectors on the chosen proxy nodes between the client and the server entity. The first-hop proxy node+selector pair is then communicated back to the client entity which uses it to initiate a session with the server through the preconfigured chain.

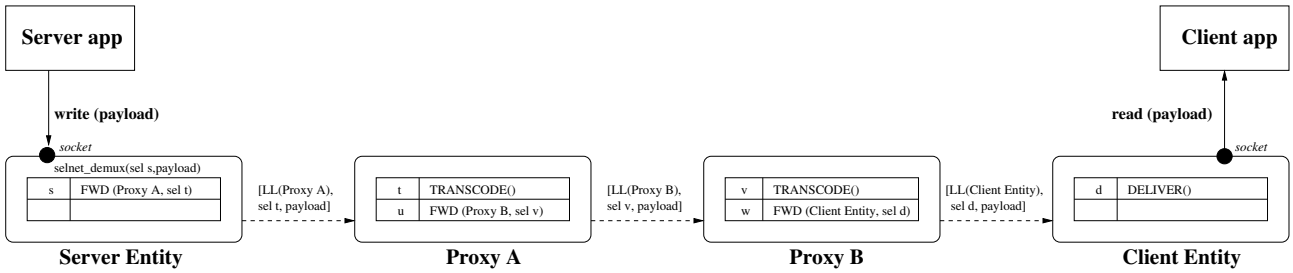

Figure 2: Content Adaptation in SelNet. (FWD=forwarding function, TRANSCODE=transcoding function, LL=link layer)

4.1 The Client Entity

The client entity is a configurable SelNet node which resides at or close[1] to the client terminal. It will sit and listen to the network traffic requested by applications on the terminal. The client entity has been configured with information about the capabilities of the terminal and a ruleset which will decide whether or not it should intercept outgoing traffic. This ruleset can be bound to activate when certain protocols, ports or hosts are accessed.

When the ruleset specifies that the client entity should intercept the traffic, it will try to set up a content adapation path through the network. This is achieved by contacting a proxy provider within the content adapation network with information about its own capabilities and which target is requested. The client entity will then proceed to wait for a reply. The reply will contain the first-hop proxy and a unique selector to be used to initiate a session with the server entity over the specially designed adaptation path. When the session has been initiated and transcoded data begins to arrive, the client entity will use its *deliver* function to pass the payload of the packets through a socket to the application.

4.2 The Proxy Provider

The main task for the proxy provider is to set up an adaptation path from the client entity, through various proxy servers, to the server entity. This path should be dynamically built to suit the different needs of the many different terminals trying to connect to the server entity. The proxy provider makes a decision based on the information provided by the client and the knowledge of the proxies in its control to create an ordered sequence of the proxies to be used for this specific session. This ordered sequence is then to be imprinted on the chosen proxy nodes. This is where the features of SelNet become very useful. By using SelNet selectors to set up the path through the network we will not only be able to define a sequence containing which proxies to invoke, in which order they should be invoked and how to invoke them on a global level. We can at the same time set up a network proxy path from the client to the server, which the client can access simply by activating the first-hop node+selector pair.

The proxy provider creates two types of selectors on each node in the proxy chain: a *proxy* selector which will be used by the client to set up the control path from the client to the server and a *transcode* selector which will be used on the return path to transcode the data on the way back to the client. When the adapation chain is initiated the proxy function describes which node+selector is next in the control path to the server and the transcode function describes which transcoding to perform and which proxy selector it is bound to. As the control path is resolved through an XRP RREQ coming from the client to the first-hop proxy, the node will respond using an XRP RREP, configure its transcoding selector to point back to the client and issue its own RREQ for the next proxy in the chain. Similarly the next proxy will point back to the previous proxy in the chain until finally the server entity is reached by an RREQ from the end proxy. On the way back the server entity will use a *forwarding* function to reach the transcoding selector on the end proxy, which in turn will forward the transcoded data to the next transcoding selector until the data reaches the client entity delivery selector.

By using its knowledge of the client terminal and its control of the different proxies in the network, the proxy provider is able to set up an optimal way through the network. It is in total control of the adaptation process in the sense that only the proxy provider has knowledge of the entire sequence of transformations taking place. Each invoked proxy will only have local information about what itself is supposed to do and its immediate neighbors.

4.3 The Server Entity

The server entity is the node on which the content server runs. When the chain has been created by the proxy provider and the client has resolved a control path traversing the proxies through the network, as discussed above, there is no need for any additional functionality other than being able to receive requests and forward data to the nodes in the network. Figure 2 shows how this forwarding and transcoding takes place on the way back to the client. This process will be explained in detail the next section.

Within the discussed framework it is quite possible to have the server entity and the proxy provider residing on the same physical location within the network. However, we feel that it would be more suitable to let the content provider run a single server entity node to respond to direct requests. The proxy provider, and the proxies under its control, would be more suited to be run by a third party with which the content provider has a service level agreement. This way the proxy provider can cater its functionality to a number of different content providers and still ensure the integrity and copyright issues important to the content provider.

5. SCENARIOS

The proxy and adaptation path through the network is initiated when the proxy provider configures the path by creating the *proxy* and *transcoding* selectors on the chosen two proxy nodes. However, the path is not activated until the client entity has resolved a control path

[1]For example a 3G phone which does not run a client entity, could have a client entity located in the 3G gateway as suggested in [1].

through the entire network to the server entity. The control path is established using XRP resolution requests (RREQ) specifically tailored to trigger the proxy selector on the nodes. Once the proxy selector is activated it will not only send the standard resolution reply, as in the example of static forwarding above, but configure its own transcoding function to point back to the client. More specifically, if a node X wants to communicate through a proxy on node Y, node X broadcasts an RREQ to the advertised XRP selector on node Y. This request specifies the address that node X wants to set up a proxy on and which selector it is set to trigger.

There could be any number of nodes in between the client and the various proxies. The SelNet network could also be partially deployed, co-existing with other network infrastructure as discussed in [3]. We are going to simplify the model scenario by ignoring the actual routing taking place between the nodes.

5.1 One Chain of Two Proxies

The first steps of the process of setting up a proxy and adaptation path through the network is shown in more detail in Figure 3. Before sending out a RREQ, the client node C installs a function at a new selector r to handle any resolution reply (RREP) and a function d to handle any incoming transcoded data for delivery of packets to its IP stack. Let us assume that proxy A is the first-hop node, which the client wants to use to set up a proxy and adaptation path through the network.

The client C will begin by sending out an XRP RREQ, carrying the target label A and the target proxy selector u (**1**). Proxy A decides it is the target of the RREQ and that it should send back a reply and at the same time configures its transcoding function to point back to C and sends out a similar RREQ to the next proxy in the chain, proxy B with selector v as the target. The RREP sent back to client C simply confirms that the proxy control path has been set up in front of C and that it is okay to begin transmitting data to the specified node+selector pair (**2**). Based on the RREP, client C will install a forwarding entry with selector f pointing to the pair A+u (**3**. Similarly proxy B will set up communication with the content entity. The first control data sent from client C through proxies A and B might be a standard HTTP GET. This is going to cause the content entity to begin forwarding data along the reverse transcoding path which was set up behind the control path as seen in Figure 2.

6. OUTLOOK

The contribution of the SelNet architecture is twofold. The architectural approach of placing indirection at the bottom of the protocol stack and the way we can maintain flexibility in the face of changing requirements by the usage of addressing packet processing functions rather than only nodes. The label switching will also make packet forwarding faster than with IP [6]. The main advantage of using SelNet as a content adaptation network is that we have an easy and flexible way of creating indirection through the network to set up an adaptation path leading from the server, through several separated proxy nodes and back to the client. And this without having to build in complicated solutions at the application layer.

We already have the core infrastructure of SelNet implemented and we have implemented an ad-hoc routing protocol called *LUNAR* using SelNet for forwarding and demultiplexing [4]. We also have a real-world implementation of a simple distributed proxy scenario,

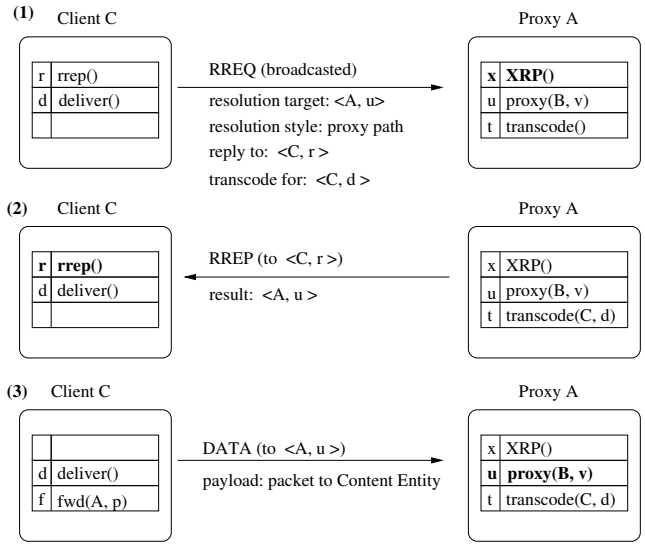

Figure 3: SelNet performing a Proxy resolution.

which we will continue to develop. There are a number of interesting proxy services which can be used in this context. We have considered the following: VJ TCP header compression, Gzip compression, RabbIT web proxy, Squid web proxy cache and Glomop/Transend image transcoding proxy. The next step will be to create a single protocol implementation of dynamic distributed content adaptation, using the framework presented in this paper.

7. REFERENCES

[1] 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).

[2] 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).

[3] GOLD, R., GUNNINGBERG, P., AND TSCHUDIN, C. A virtualized link layer with support for indirection. http://user.it.uu.se/ rmg/pub/fdna05-gold.pdf.

[4] 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).

[5] 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.

[6] 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).