A proposed architecture for integrating Active Networks and MPLS

Sanda Dragos & Martin Collier
Dublin City University, School of Electronic Engineering
Dublin , Ireland
e_mail: {dragoss, collierm}@eeng.dcu.ie

Abstract: **Multiprotocol Label Switching (MPLS) was originally intended as a fast switching technique for core networks, but proved to be very useful for providing traffic engineering and QoS Routing capabilities in the Internet.**

Hence, MPLS is beginning to appear at the edge of networks as well as in the network core. Modern edge routers are typically provided with a multi-layer switching capability to support packet computation services within the network, whereas MPLS employs layer two switching. This paper describes a method to emulate the capabilities of multi-layer switches in MPLS by combining it with the active network concept.

Keywords: MPLS, Active Networks, access networks, Linux

1 Introduction

The main function of a network is to deliver packets from one end-point to another. However, there are situations, mainly in access areas, where packets need to be processed within the network. This is the case of firewalls, Web proxies, multicast routers, mobile routers or other similar services that need to access information from a packet header to decide whether the current packet should be dropped or how it should be forwarded. More generally, the packet header or data may be modified.

As MPLS evolves beyond the core network, penetrating the access area, a big issue that arises is its inability to perform packet computations for services such as the ones described above. Active Networks is a novel solution for implementing such services providing a flexible network infrastructure with increased capabilities.

In this paper, we propose the integration of Multiprotocol Label Switching with the Active Network concept as a solution for the future access network. The resulting network will provide all the features of MPLS, and in addition, will use active packets to support packet processing.

An overview of MPLS and Active Networks and the motivation for integrating them is presented in section 2. In section 3 we propose an architecture for such an integration, and section 4 describes our implementation which proves that such an integration is possible.

2 Background

2.1 MPLS Overview

MPLS [1, 2, 3] is a packet label-based switching technique which was originally devised to perform fast switching in the core of the network. The argument for that was that the algorithm for the "*longest-prefix match*" is more complex and time consuming than the exact match used by MPLS, where the label is an index into a table.

In time, MPLS proved to have other qualities even more appealing than fast switching [4]. Being a relative simple connection-oriented protocol, it proved to be suitable for implementing traffic engineering and Quality of Service (QoS) Routing in a simpler way than using IP.

The packet headers (transport layer (TCP) header and network layer (IP) header) is analyzed only once when the packet enters the MPLS network, to determine its label. Switching decisions thereafter are based on this label - the next MPLS node will inspect this label to decide what to do with the packet: whether to pop the label, push another label or to swap the label and to forward the packet to another MPLS node.

The desire to provide the QoS capabilities of MPLS end-to-end motivates its deployment not only in the core network but in the access network as well. Thus, many now advocate the use of MPLS networks in the access area [5].

However, a disadvantage of MPLS is that the forwarding process is opaque in the sense that the network is insensitive to the packets it carries and they are transferred between MPLS nodes without modification. In the access area there are situations where such modifications are required.

In today's networks, there are two ways of implementing such computations within the network:

In the traditional ("passive") way, the processing within the network is limited primarily to routing, congestion control and QoS schemes and the points that perform such computations are fixed elements in the network. Such a network poses difficulties in integrating new technologies and standards into the shared network infrastructure, in reducing redundant operations at several protocol layers and in accommodating new services in the existing architectural model.

A innovative idea was proposed in [6] as a solution to this problems: to give the user the possibility to program the network. This technology is called *Active Networks* [7, 8, 9].

2.2 Active Networks

Active networks are "active" in two ways: nodes within the network can perform computations on packets traversing them; and users can "program" the network by supplying their own programs to perform such computations [7]. There are three architecture approaches for implementing active networks. They differ by the placement of the code in the network.

2.2.1 The Active Nodes Architecture

The authors of [7, 8] refer to this as the *discrete model* or *Programmable switches* approach because the programs and data are carried separately (i.e., are discrete), while [9] refers to it as the *active node approach*.

In this approach, the packets carry some identifiers or references to predefined functions that reside in the active nodes. The packets are active in the sense that they decide which functions are going to be executed on their data, but the actual code resides in the active node.

2.2.2 The Active Packets Architecture

The Active packets approach, referred to as *the integrated approach* or *capsules* in [7, 8] is characterized by the fact that the code is carried by the packet. The nodes are also active because they allow computations up to the application layer to take place, but no active code resides in them. This is the reason why [9] refers to this approach as *active packets*.

Being carried by the packet the code within an active node cannot be very large, but only composed of "primitive" instructions, that perform basic computations on the packet's content.

2.2.3 The Active Packets and Active Nodes Architecture

The third approach is a combination of the previous two, where active packets carry the actual code and other more complex code resides in active nodes. Using this approach, users can choose either the active packets approach or the active nodes approach according to the nature of their application.

2.3 Terminology in MPLS

The main elements of an MPLS network are the MPLS nodes/routers which are called *Label Switching Routers*(LSRs). They forward the packets within an MPLS network. The routers at the edge of the MPLS domain are called *Label Edge Routers*(LERs) and they examine the packets headers and based on the information within those headers the packet is assigned to a class called *Forwarding Equivalence Class*(FEC). All packets within the same Forwarding Equivalence Class will be labelled with the same label and will be treated equally in the network.

Note: We will refer later on in the article to an MPLS node with active capabilities as an "*active LSR*".

3 MPLS and Active Networks

Given the arguments in the previous section we propose the integration of MPLS with Active Networks as a possible solution for the future access networks.

Figure 1 illustrates how the routers of an IP network can be augmented to have active capabilities. These active routers can coexist and interoperate with legacy IP routers, which transparently forward packets in a traditional manner.

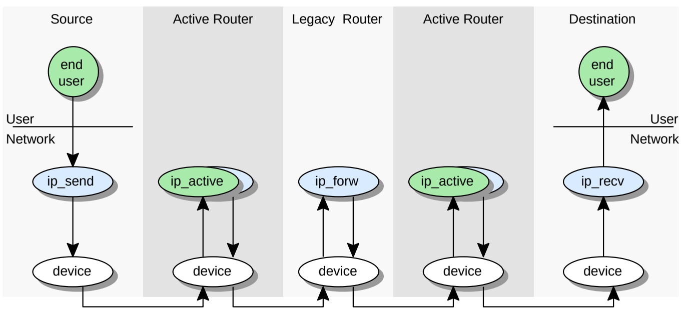

Figure 1: Packet processing within the nodes of a legacy Active Network

In Figure 2 we propose a network in which some LSRs are augmented with active capabilities. Legacy LSRs and active ones can coexist and interoperate.

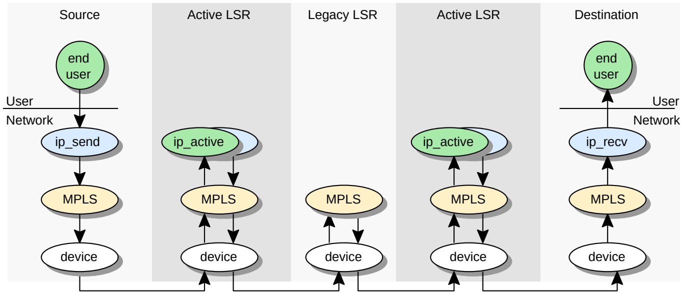

Figure 2: Packet processing within the nodes of an MPLS Active Network

A more detailed architecture of an active LSR is shown in Figure 3. A packet is determined to be an "active" packet or not based on the label. If the packet is "active", it is sent to the IP layer and from there to the relevant active code in order to be processed. After being processed, the packet is sent back to MPLS as if it were generated locally. Then, the packet is assigned to a Forwarding Equivalence Class and labelled corresponding with its class.

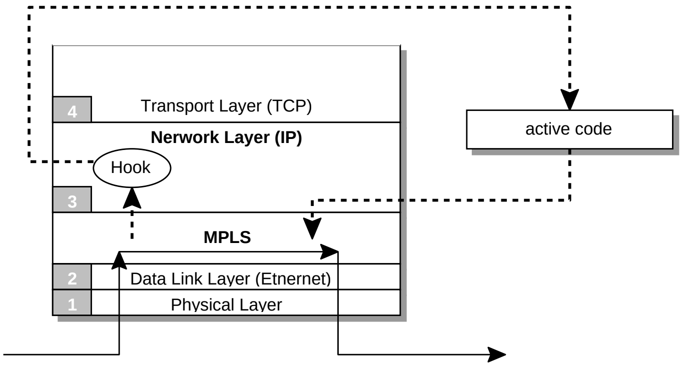

Figure 3: An active MPLS architecture

Using this architecture only the active packets (as identified by the label) that traverse an active LSR will be processed by the active code. Conventional packets will be processed in accordance with the standard MPLS protocol.

4 IMPLEMENTATION

To prove the concept of integrating MPLS and active networks, we set up a minimal MPLS network using the mpls-linux implementation [10] as shown in Figure 4. We set up the labels and we sent packets from **LSR A** to **LSR C** through **LSR B**.

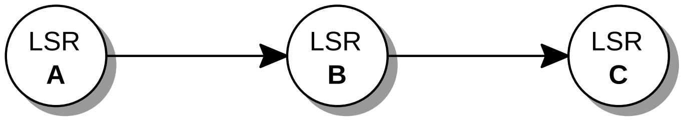

Figure 4: A minimal MPLS network

We set labels so that active packets that travel from **LSR A** to **LSR C**, arriving at **LSR B** are identified by a certain label and consequently they will be sent up to the IP layer. Then, using the netfilter framework, we implemented an trivial example of modifying ("mangling") the packet.

Netfilter [11, 12, 13] is a part of the networking software within the Linux kernel. This framework offers the possibility of "packet mangling" (i.e. modification of the header or payload contents). For each networking protocol netfilter implements "hooks" that are well defined points in a packet's traversal of that protocol stack. In IPv4 there are 5 such "hooks" defined as illustrated in figure 5:

1. NF_IP_PRE_ROUTING
2. NF_IP_LOCAL_IN
3. NF_IP_FORWARD
4. NF_IP_POST_ROUTING
5. NF_IP_LOCAL_OUT

Parts of the kernel can register to listen to the different hooks for each protocol. If someone has registered for a particular protocol and hook, the packets passing that point in the protocol stack can be directed to the netfilter framework, where they can be dropped (NF_DROP), accepted (NF_ACCEPT) or queued to userspace (NF_QUEUE).

Queued packets are sent to userspace, where a userspace process can examine the packet, can alter it, and reinject it at the same hook from which it left the kernel.

The packets could also be processed directly in the kernel. However, the workload of the kernel would be increased, which would adversely affect the performance of the operating system. Also, packets arriving after the active packet would be delayed while it was being processed. By processing the active packet in userspace, conventional packets that succeed it encounter no such delay, and the active packet is re-injected asynchronous (if appropriate) into the packet stream after processing.

Using the netfilter framework, we registered to listen the first hook defined by IPv4 (NF_IP_PRE_ROUTING) which is placed at the entry of the packet in the protocol stack, just after the sanity checks (i.e., not truncate, IP checksum OK, not promiscuous receive). We captured packets passing that

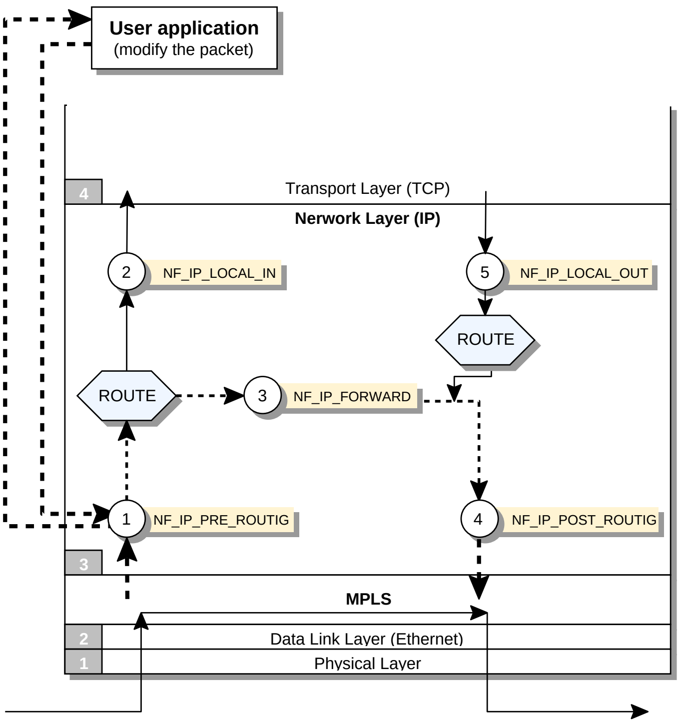

Figure 5: A MPLS node modified to programme the packets

hook and queued them for userspace. We wrote an userspace application which modifies the packet's source address. We tested the system as you can see in Figure 6, by sending ICMP (Internet Control Message Protocol) messages from **LSR A** to **LSR C**.

A response ICMP message should be received by **LSR**

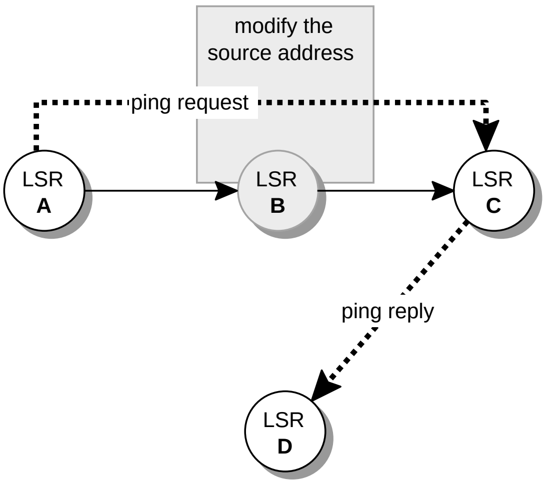

Figure 6: Our experiment test

A as a reply for each ICMP message sent, but because packets from **LSR A** destined to **LSR C** went through **LSR B**, their source address was changed with the address of **LSR D**. Therefore, **LSR C** received the ICMP messages but saw that the packets came from another destination (as the destination was altered) and sent the reply to that destination, which was **LSR D**.

Although this implementation it is only a *"proof of concept"*, it wants to simulate an active MPLS network in which we considered **LSR B** to be an "active" LSR, therefore it should sent the packets received with a specific label to IP protocol stack. From IP stack we captured the packet and we sent it to an application that modifies the packet (e.g the source address). The modified packet was then labelled and sent to the next hop.

In a legacy LSR a transit packet will go up to the MPLS layer where, based on the label, the packet is forwarded to the next hop.

In our proposed active LSR, packets arriving with a certain label which specifies that they are "active" are sent up to network layer (IP) and from there to an active code.

This experiment, whilst trivial, demonstrates the principle used to provide active packet support in MPLS networks. The benefits of providing such support depend on the capabilities of the active code and its interaction with the MPLS software. We are currently developing an interface between the linux-mpls implementation and our active code modules, which will enable active packets to be used to fine-tune the operation of MPLS so as to emulate the features of multi-layer switching technologies. This features can be used in solving problems such as MPLS based Web switching load balancing [14].

5 Conclusions

We propose in this article the integration of two edge technologies, the Multiprotocol Label Switching and Active Networks. Integrating these technologies overcomes a significant limitation of MPLS in access networks, namely its inability to perform switching above layer two. We have implemented a prototype network using Linux, which proves the validity of the concept. Future work will address how to exploit the flexibility of the active MPLS concept to address tasks such as firewalling without requiring inspection of the IP or TCP headers.

References

[1] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. Technical Report RFC3031, IETF, 2001.

[2] Eric W. Gray. *MPLS Implementing the technology*. Addison-Wesley, 2001.

[3] Uyless Black. *MPLS and Label Switching Networks*. Prentice Hall, 2001.

[4] Sanda Dragos, Radu Dragos, and Martin Collier. Bandwidth management in mpls networks, 2001. http://telecoms.eeng.dcu.ie/symposium/papers/B1.pdf.

[5] Guy Chenard. Mpls in the access: Creating differentiated cos for ip-based services. http://www.mplsworld.com/archi_news/itv/itv_10_gchenard.htm.

[6] Active networks. http://www.darpa.mil/ato/programs/activenetworks/actnet.htm.

[7] D. L. Tennenhouse and D. J. Wetherall. Towards an active network architecture. *Computer Communication Review*, 26(2), April 1996.

[8] David L. Tennenhouse, Jonathan M. Smith, W. David Sincoskie, David J. Wetherall, and Gary J. Minden. A survey of active network research. *IEEE Communications Magazine*, 35(1):80–86, 1997.

[9] Konstantinos Psounis. Active networks: Applications, security, safety, and architectures. *IEEE Communications Surveys*, pages 1–16, First Quarter 1999.

[10] Mpls for linux project. http://mpls-linux.sourceforge.net/.

[11] Rusty Russell. Linux netfilter hacking howto. http://netfilter.samba.org/unreliable-guides/netfilter-hacking-HOWTO/index.html.

[12] Rusty Russell. Netfilter: Packet mangling in 2.4. September 1999. Augsburg.

[13] Rusty Russell. Writing a module for netfilter. *Linux Magazine*, June 2000.

[14] Radu Dragos, Sanda Dragos, and Martin Collier. Design and implementation of an mpls based load balancing architecture for web switching. to be published on the 15th ITC Specialist Seminar: Internet Traffic Engineering and Traffic Management, 2002.