High Router Flexibility and Performance by Combining Dedicated Lookup Hardware (IFT[1]), off the Shelf Switches and Linux

Christian Duret[1], Francis Rischette[1], Joël Lattmann[1], Valéry Laspreses[1], Pim Van Heuven[2], Steven Van den Berghe[2] and Piet Demeester[2]

[1] France Telecom R&D, Issy les Moulineaux, France
{christian.duret, francis.rischette, joel.lattmann, valery.laspreses}@francetelecom.com
[2] IMEC, Ghent, Belgium
{pim.vanheuven, svdberg, demeester}@intec.rug.ac.be

Abstract. In this paper we propose a new router architecture that combines both flexibility and performance. This router architecture aims at combining the best of two worlds: the commercial routers, which have a proven track for stability and performance but lack the flexibility of routers with open source operation system. The latter is particularly flexible because the source code is accessible for analysis and modification purposes as opposed to the traditional commercial routers, whose software can be altered by their manufacturers only.

1 Motivation and State-of-the-Art

The exponential growth of Internet traffic has yielded a dramatic development effort of the IP routers technology. Moreover, the deployment of value-added IP service offerings (ranging from a QoS-based access to the Internet to real-time services, like IP videoconferencing) has lead to an important development of specific capabilities (traffic conditioning, marking, scheduling and metering) that are supported by some - if not all - the routers of the Internet. The consequence of the activation of such enhanced capabilities is twofold: a demand for an increase of the routers' switching performances together with the availability of multi-functional and multi-service routers.

Other important concerns deal with IP security, multicast, and Virtual Private Networks services. Therefore, the IP routers that are exploited in a multi-service environment need to be flexible enough in order to address current and future requirements.

For the past decade, Linux has received considerable interest not only from the research community, but also from the industry. An extensive description of Linux features and related bibliography can be found in [1]. Recently, an implementation for DiffServ over MPLS [2] has been released by some of the authors of this paper.

[1] IP Fast Translator

The main issue raised by the use of Linux-based routers deals with their switching and forwarding performances:

- They are bounded by the CPU and are difficult to predict since both the data and the control planes run on the same CPU;
- Another problem is the interrupt overhead. Note that alternatives exist which are based on polling [3].
- Even more important is that most of these routers are built around commodity PC, and therefore inherit of their shared bus limitations;

Commercial routers provide more than acceptable switching performances. Their main drawback is their lack of flexibility. Thus, whenever an IETF standard is not implemented yet, and/or some functionality is missing, it becomes necessary either to rely on the roadmap of a given manufacturer for the introduction of new features, or to add adaptation boxes, where it is feasible.

The commercial routers whose architecture is based upon a high performance CPU and interface cards linked together by a shared bus, are not sufficient anymore to keep pace of the constant increase of Internet traffic, hence overwhelming the Moore's law. A new class of components, dedicated to high speed network layer processing has emerged for about a year: the network processor. Unfortunately, network processors are clearly designed in an opposite way as the Linux paradigm.

2 The IFT-Based Experimental Router

Several years ago, FTR&D has developed a research program on high speed networking techniques to be initially deployed within an ATM context, so as to address the above-mentioned issues. One way to address the switching performances issue consists in system optimization. Looking at a conventional router, one can see that less than 5% of the system software runs in the data path, but is responsible for more than 95% of the execution time. Only a small part of the related functions has to be "wired" to reach the performance level that is needed today, this level being around 1.5×10^6 packets/second per Gigabit/s bit rate at the interface level. Among these functions, classification ("The process by which a data packet is examined and policy decision are made which affect down-stream processing" [4]) is a critical one, and it clearly requires as much flexibility as does a purely software-based implementation to handle forwarding decisions, filtering such as Access Control Lists (ACL), an increasing set of encapsulations headers, forthcoming protocols (IPv6)...

Generally speaking, the incoming frame is characterized by a set of fields within a succession of headers, whose respective contents could possibly be analyzed against a set of patterns. Each individual analysis is defined by the position of the field within the frame, the set of patterns against which to compare the content of the field, an action to be performed in the case of a match (either a link that leads to another field to be analyzed, or a final result that indicates where to send the packet, or a default treatment). Figure 1 below is an example of such behavior for basic IP forwarding.

The implemented lookup process is basically a "multibit Trie" allowing for either exact range or longest match. An extensive survey of lookup algorithms can be found in [5]. The complexities reported for this lookup scheme are:

- Worst case lookup time O(W)
- Worst case update time O(W/K + 2K)
- Worst case memory size O(2kNW/K)

Where N is the number of entries, W the length of the address and K the size of the bit slice (or "stride" according to [5]).

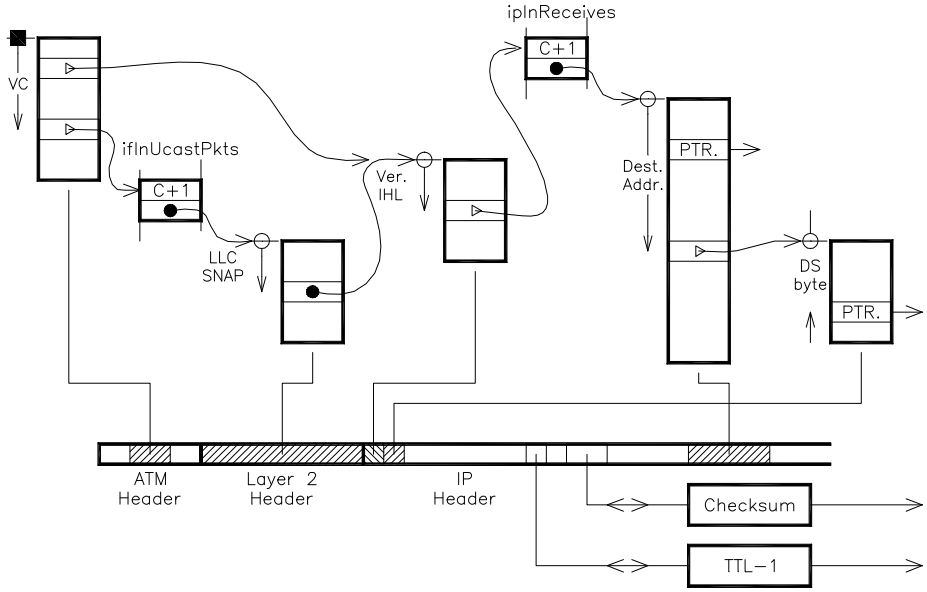

Fig. 1. Successive header fields to be processed for basic IPv4 forwarding.
The shaded areas within the incoming frame are the header fields that are analyzed through the IFT. The sequence of the analyzed fields and related counters update is fully defined by the pattern store memory that implements a finite state machine, whose transitions are triggered by the incoming packet (upper part of the figure). A match may also trigger external processes to keep track of layer succession, check the header, update counters, update checksum, Time To Live (TTL) and Differentiated Service Code Point (DSCP). The IFT analysis result is mapped onto a VCI (Virtual Channel Identifier) value that implicitly designates the output port. These processes depicted in the lower part of the figure are protocol-dependent.

The worst case lookup time is 120ns for IPv4 addresses in the present hardware implementation, to be compared to 2.99us reported in [5] for a software implementation executed on a 200 MHz Pentium-Pro based computer under Linux.

By nature, there is neither layer nor any field restriction in the analysis: upper layers may be processed through linked tables. The worst-case lookup time is 345ns for a basic TCP-UDP/IP 5-tuple, to be added to regular forwarding process time. This classification is performed by implementing a "set-pruning trie" data structure according to the proposed taxonomy in a recent survey of algorithms for packet classification [6]. The properties of this structure are:

- Worst case lookup time O(dW)
- Worst case memory size O(dN)

Where d is the "dimension" of the classifier, that is to say the number of header fields of W bit length on which a number N of classification rules apply. The large amount of memory is due to the fact that some fields may need as much as dN tables to ensure that every matching rule for a given field will be traversed depending upon the result of the analysis of the previous field. No backtracking nor linear search are needed allowing to analyze each relevant field only once on the fly.

Backtracking, as implemented in "Grid-of-tries" [7], reduces the storage complexity to O(NdW) at the expense of O(Wd-1) for worst-case lookup time complexity.

Incoming packets are analyzed at line rate by reading the IFT control memory. A software driver running on the IFT host is in charge of writing it. This driver offers a set of updating functions: insertion and removal of patterns. It constantly provides the global consistency of the control memory, without the need of recurrent tables reorganization. The IFT driver runs on a logical copy of the IFT memory and performs incremental updates, thus the memory bandwidth required for update operations is several orders of magnitude lower than the bandwidth required by incoming packet processing.

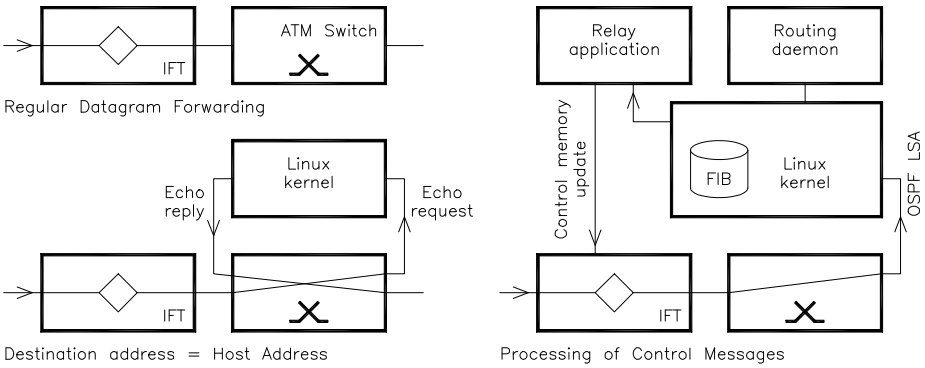

Fig. 2. Examples of packet processing.
IFT-only functionality: The IFT runs a copy of the kernel Forwarding Information Base (FIB). A datagram whose destination address has been recognized is forwarded directly by the IFT to the switch fabric where it is forwarded to the output interface that leads to the next hop associated to the contents of the destination address field of the datagram.
Linux control path functionality: a datagram destined to the router is forwarded to the router Linux host. The Linux kernel then processes this datagram. For example, if it is an Internet Control Message Protocol (ICMP), Echo Request message, then the kernel sends an Echo Reply message back to the originating host through the switch fabric.
Linux control and IFT configuration functionality: a datagram destined to the router is forwarded towards the router Linux host. The Linux kernel sends this datagram up to the application layer. For example an Open Shortest Path First (OSPF), Link State Advertisement (LSA) packet is sent to the routing daemon. The daemon will update the kernel FIB if needed. The corresponding message is then copied in the IFT control memory.

The communication within the IFT-based experimental router is performed through an ATM switch fabric that directs IFT-processed packets towards external (most of packets) or internal interfaces for being handled by the Linux host and the

control plane processes. Thus, aside the IFT driver, the role of the Linux host is threefold:

- In the data plane, processing of the datagrams that were sent to the Linux kernel by the IFT module (such as time exceeded ones or those containing options or directly addressed to the router);
- Running the control plane functionality;
- Configuring the IFT forwarding table through a user relay application.

The control path functionality is comparable to a regular Linux-based router. Figure 2 gives the three possible scenarios that can occur when a packet enters the experimental router. Routing protocol packets are an important example because these packets can update the routing table inside the Linux component. These changes have to be reflected in the IFT forwarding table too. This leads to the third role of the Linux components: the configuration tasks that consist of mapping the Traffic Classifier configuration commands and routing updates using netlink sockets [8] onto IFT header pattern entries. This has the advantage that software-based routing daemons can be re-used on the experimental platform without the need for any modifications.

3 Future Work

The present router design is based upon an ATM switch. Ongoing developments include the support of Fast and Gigabit Ethernet interfaces. The architecture described in this paper applies to a design based upon an Ethernet switch as well. In this case, the IFT analysis result, instead of being mapped to an ATM connection, is mapped onto a Medium Access Control (MAC) frame, whose Destination Address field is either a host, a gateway or the Linux host itself. Additionally, most of Gigabit Ethernet switches provide priority queuing mechanisms through the implementation of the IEEE 802.1p standard that may be useful for implementing Diffserv-based routing and QoS mechanisms.

The IFT developments have been considered for the implementation of a Multimedia Switch Router [9]. Security applications are also considered [10]. Another application of this kind of platform could be admission control facilities that would be based upon "on the fly" identification of elastic and streaming flows [11].

4 Conclusion

In this paper we explained that current marked trends push for both flexible and high performance routers. Current router options are either high performance (commercial routers) or flexible (open source-based PC routers).

As a solution to this problem, we propose a router architecture that consists of the combination of fast dedicated look-up hardware, off-the-shelf switches, and the Linux OS. The combination of these components provides:

- A performance level that can easily be compared to the switching performances of commercial routers;
- Scalability through the use of off-the-shelf switching fabric (currently ATM, Fast and Gigabit Ethernet later on);
- The flexibility at the control path equal to that of an open source PC router;
- The extensive developer support that have been engaged on Linux-based routers;
- A clear separation between forwarding and control planes.

Acknowledgements

This work was partly undertaken in the Information Society Technologies (IST) TEQUILA project, partially funded by the Commission of the European Union. Part of the work is also sponsored by the Flemish Government through two IWT scholarships. The authors would also like to thank the rest of the Tequila colleagues who have contributed to the ideas presented in this paper and Jacques Le Moal, Jean Louis Simon (France Telecom R&D) for their contribution to the IFT project.

References

[1] D. Griffin editor “D2.1: Selection of Simulators, Network Elements and Development Environment and Specification of Enhancements" http://www.ist-tequila.org

[2] Pim Van Heuven, Steven Van den Berghe, Tom Aernoudt, Piet Demeester, "RSVP-TE daemon for DiffServ over MPLS under Linux", http://dsmpls.atlantis.rug.ac.be

[3] Benjie Chen et. al.," The Click Modular Router Project", http://www.pdos.lcs.mit.edu/click/

[4] "Programming & Reprogramming: Keeping the speed without Losing your Mind" in Network Processor Summit - Networld+Interop 2000

[5] Miguel A. Ruiz-Sanchez, Ernst W. Biersack, Walid Dabbous "Survey and Taxonomy of IP Address Lookup Algorithms" in IEEE Network March/April 2001

[6] Pankaj Gupta, Nick McKeown "Algorithms for Packet Classification" in IEEE Network March/April 2001

[7] V. Srinivasan et al., "Fast and Scalable Layer four Switching" in Proc. ACM Sigcomm, Sept. 1998

[8] G. Dhandapani, A. Sundaresan “Netlink Sockets – Overview” http://qos.ittc.ukans.edu/netlink/html/

[9] Michel Accarion, Christophe Boscher, Christian Duret, Joël Lattmann "Extensive Packet Header Lookup at Gb/s Speed for an Application to IP/ATM multimedia switch router" In World Telecommunication Congress - International Switching Symposium, Birmingham May 2000

[10] Olivier Paul, Maryline Laurent, Sylvain Gombault, "A Full Bandwidth ATM Firewall" in Proc. of the 6th European Symposium on Research in Computer Security, Toulouse, France, October2000

[11] N. Benjameur, S. Ben Fredj, S. Ouslati-Boulahia, J. Roberts, "Integrated Admission Control for Streaming and Elastic Traffic" in M. Smirnov, J. Crowcroft, J. Roberts, F. Boavida (Eds), "Quality of Future Internet Services", Springer, LNCS 2156, 2001.