CS 636 Final Project Report
XPRO IP Router

Ahsan Habib   Radu Sion   Jamie Van Randwyk
Adam Welc
(BING6)

April 30, 2001

1 Notes on our implementation

We used almost no code as well as almost no design ideas from Volume 2 (with the exception of ARP as noted). We managed our code using CVS. We designed a portable debugging facility that can be disabled easily but allowed for easy testing.

2 Routing structures

The routing structures are designed to maximize concurrent access and to minimize lookup time. We made several reasonable assumptions as to the structure and associated sizes and network patterns.

The number of concurrent directly connected routers is less than 255. The update rate to the routing structures is much less than the lookup rate. Thus we aim at optimizing lookup time even in the detriment of higher update times. The entire routing process honors longest prefix and CIDR assumptions.

2.1 Prefix clustering

A number of papers (see bellow) on routing and lookup structures suggest that current trends in addressing, especially considering CIDR (see Mae-West, 1996 table statistics), lead to certain clustering patterns in the routing prefixes. This clustering characteristic can be exploited successfully by focusing on the hot spots within the routing prefixes and using fast direct lookup hashes.

2.2 Routing cache and hashing

The routing structures include a small fast cache (3 slots) that is implemented using a direct hash and is first consulted, before any other lookup operation is done.

Many operations require serial access to all (or most) of the routes in the router at a certain point in time. This O(n) processing can be sometimes avoided by maintaining separate linked lists (ex. RIP advertising of dead routes) but in other cases this cannot be easily and efficiently done. Thus a O(n) (hashtable with linked lists) structure is maintained inside the router. This structure is updated at the same time as the other (lookup) structures but allows serial access to the entire table without locking any parts of the optimized lookup hashes.

2.3 Main routing data structure – TRIX

The main lookup mechanism relies on a novel structure we called TRIX. The structure combines hashing with an LC Trie. The main idea is to exploit prefix clustering extensively but at the same time to offer a good lookup time.

The structure is mainly a hierarchy of prefix hashes. Depending on the statistical properties of the prefix clustering lookup times can be reduced greatly sometimes with over 83% of lookups done in just one memory access (one hash lookup). This, combined with the existence of the routing cache provides a powerful and fast lookup mechanism.

We didn't have time to finalize more statistical measurement but we plan on using data from existing sample industry level routing tables to come up with more accurate numbers and even argue many advantages over LC Trie approaches.

Depending on the hashing factors and the statistical properties of the routing prefixes, the TRIX size can vary greatly and it is true that in the worst case scenario (no prefix clustering whatsoever) it degenerates into a LC Trie.

But in most cases it can easily fit into a 512-1MB processor cache, allowing for fast access.

2.4 Concurrent access

Synchronization being paramount for the good function of the router, we have to make sure that lookup occurs smoothly and lookup is not slowed down by updates and other lookups more than necessary. We implemented thus a pointer switching and flushing strategy within the TRIX structure that uses two underlying core structures that are updated alternatively, the lookup being performed always from the passive (non-updated) structure with a maximal level of concurrency.

3 Additional functionality

We implemented several additional features extending basic functionality of a router.

3.1 NAT

We implemented the Network Address Port Translation (NAPT) variation of NAT. Our version of NAT handles UDP and ICMP messages only. (i.e. NAT would have to be modified if TCP were added.). It translates all possible combinations between UDP and ICMP. For udpecho type of application, we need UDP-UDP translation. For traceroute using UDP datagram, a host sends UDP datagram but gets ICMP time-exceed message back. NAT needs to remember the UDP state and translates properly when it gets the corresponding ICMP message. Again traceroute using ICMP sends ICMP echo request and gets ICMP time-exceed and our NAT takes of proper translation.

3.2 IGMP

We implemented IGMP version 1 and our router is a multicast router. A host can dynamically join and leave a multicast group. Our multicast router keeps state information about hosts listening to a particular group and forwards multicast packet based on this information. The multicast router also periodically sends query to all the networks it is connected with and if it does not get any response for 3 (from RFC 1075) queries it stops forwarding packets to the network. The multicast router has to do this because in version 1, host does not report when it leaves a group.

3.3 Firewall

We implemented a mechanism to filter IP packets before their processing by IP module. We support multiple, nested filters as well as adding, enabling, disabling, deleting filters and printing detailed information about filters available in the system.

4 Applications

- UDP echo – both client and server have been implemented
- ping – IP options (record route and timestamp) can be used while pinging remote machines
- traceroute – traceroute has been implemented to use either UDP datagrams or ICMP messages while querying remote machines.
- multicast chat – application for testing and demonstrating the multicast functionality (IGMP and multicast packets forwarding); hosts can join/leave groups and send/receive messages.

5 Extensions of the required functionality

5.1 IP

We implemented IP options processing. Both processing and creation of IP options is supported. Supported options include: record route, source route (both strict and loose), timestamp. For IP reassembly, we implemented the algorithm proposed in RFC 815.

5.2 UDP

Our UDP implementation includes an improved interface. It allows for binding specific IP addresses to specific ports giving greater flexibility in writing UDP applications. We modified necessary files to support this feature. We also designed an interface between UDP and IP layers that allows for passing IP options in both directions.

5.3 ICMP

Our ICMP implementation allows for passing ICMP messages to the application layer. Also passing IP options between IP and ICMP layers (in both directions) is possible.

5.4 RIP

We implemented RIP in compliance to the RFCs. We support RIP 1 and 2 and only generate RIP 2 updates. Poison reverse is also implemented. We designed a simple mechanism for detection of netmasks in case of receiving RIP 1 updates which we choose not to ignore (because we only get RIP 1 updates from cisco5).

We also implemented the ability to filter RIP updates using a simple access list mechanism to enforce chosen policies. There are commands to enable/disable this facility and to add/remove trusted/non-trusted gateways to/from the corresponding lists.

6 Various facilities

We implemented several additional facilities, making the router more appealing and easy to use and control.

6.1 Control center

The control center allows the manipulation of routing tables, gateway lists, runtime variables, RIP processes (forcing of rip updates etc.), traffic shaper, etc.

The generic syntax of the control center is: control ¡command¿ where command is one of the following:

```
-------------------------------------------------------------------------
help                                     Displays this message
route [list| add| delete| flush| lookup] Routing table manipulation
gw [list| add ]                          Gateways list manipulation
arp [list| add| delete| flush]           ARP manipulation
rip [allow| deny| list| update]          RIP Filters and updates control
log [display| clear| level]              Log facility manipulation
vars [set| get| list]                    Dynamic variables manipulation
info                                     Displays system info
-------------------------------------------------------------------------
```

6.2 Selective traffic shaping

We implemented the ability to shape the incoming traffic based on type of packet. We can specify the number of maximum packets per second accepted for processing. Currently we can shape IP, ARP, RIP, UDP, ICMP and IGMP traffic.

This facility in conjunction with the firewall and RIP filtering (see bellow) makes a powerful protection tool against ill behaved routers and hosts on the network. (very welcome during the testing phase when the network was flooded with bad frames etc)

6.3 Runtime logging

We implement a runtime log that allows storage of log messages from all the modules, together with associated log priority levels. The log can be consulted at any time and allows for easy configuration and network debugging.

6.4 Runtime dynamic variables access

A way of modifying (synchronously) variables at runtime has been designed, allowing for much flexibility in setting various flags and values through a unified and simple interface. ("control vars set ¡variable-name¿ ¡variable-value¿")

6.5 RIP filtering

As described above, we implemented a access control facility for RIP updates and trusted/non-trusted gateways. This proved very useful while testing when many errand routers were on the network.

6.6 DNS and time service

Translation between IP addresses and domain names as well as time service (querying a time server for current time) has been enabled (both on gateway and hosts on all subnets).

References

[1] V. Srinivasan, G. Varghese, "Faster IP Lookups using Controlled Prefix Expansion", Department of Computer Science, WUSTL , 1998

[2] Theppitak Karoonboonyanan, "An Implementation of Double-Array Trie", http://www.links.nectec.or.th/ thep/datrie

[3] S. Nilsson, M. Tikkanen, "Implementing a dynamic compressed trie", Proc. WAE 98, Ed. Kurt Mehlhorn, pp. 25-36, 1998

[4] Stefan Nilsson, Gunnar Karlsson, "Fast address lookup for Internet routers", Proceedings of "Algorithms and Experiments", 1998, pp 9-18

[5] M. Waldvogel, G. Varghese, J. Turner, B. Plattner, "Scalable High Speed IP Routing Lookups", 1997

[6] P. Crescenzi, L. Dardini, R. Grossi, "IP Address Lookup Made Fast and Simple", Universita di Pisa, Dipartimento di Informatica, Technical Report TR-99-01, January 17, 1999

[7] M. Degermark, A. Brodnik, S. Carlsson, S. Pink, "Small Forwarding Tables for Fast Routing Lookups", Department of Computer Science and Electrical Engineering, Lulea University of Technology, 1997

[8] Topic-specific RFCs