Forwarding Architecture

Brighten Godfrey
CS 538 September 13 2011
Partridge: 50 Gb/sec router

A fast IP router

Good exhibition of the guts of a router and problems to be solved in router architecture
Inside the router

A. Design Summary

A simplified outline of the MGR design is shown in Fig. 1, which illustrates the data processing path for a stream of packets entering from the line card on the left and exiting from the line card on the right. The MGR consists of multiple line cards (each supporting one or more network interfaces) and forwarding engine cards, all plugged into a high-speed switch. When a packet arrives at a line card, its header is removed and passed through the switch to a forwarding engine. (The remainder of the packet remains on the inbound line card). The forwarding engine reads the header to determine how to forward the packet and then updates the header and its forwarding instructions back to the inbound line card. The inbound line card integrates the new header with the rest of the packet and sends the entire packet to the outbound line card for transmission.

Not shown in Fig. 1 but an important piece of the MGR is a control processor, called the network processor, that provides basic management functions such as link up/down management and generation of forwarding engine routing tables for the router.

B. Major Innovations

There are five novel elements of this design. This section briefly presents the innovations. More detailed discussions, when needed, can be found in the sections following.

First, each forwarding engine has a complete set of the routing tables. Historically, routers have kept a central master routing table and the satellite processors each keep only a modest cache of recently used routes. If a route was not in a satellite processor's cache, it would request the relevant route from the central table. At high speeds, the central table can easily become a bottleneck because the cost of retrieving a route from the central table is many times (as much as 1000 times) more expensive than actually processing the packet header. So the solution is to push the routing tables down into each forwarding engine. Since the forwarding engines only require a summary of the data in the route (in particular, next hop information), their copies of the routing table, called forwarding tables, can be very small (as little as 100 kB for about 50k routes [6]).

Second, the design uses a switched backplane. Until very recently, the standard router used a shared bus rather than a switched backplane. However, to go fast, one really needs the parallelism of a switch. Our particular switch was custom designed to meet the needs of an Internet protocol (IP) router.

Third, the design places forwarding engines on boards distinct from line cards. Historically, forwarding processors have been placed on the line cards. We chose to separate them for several reasons. One reason was expediency; we were not sure if we had enough board real estate to fit both forwarding engine functionality and line card functions on the target card size. Another set of reasons involves flexibility. There are well-known industry cases of router designers crippling their routers by putting too weak a processor on the line card, and effectively throttling the line card’s interfaces to the processor’s speed. Rather than risk this mistake, we built the fastest forwarding engine we could and allowed as many (or few) interfaces as is appropriate to share the use of the forwarding engine. This decision had the additional benefit of making support for virtual private networks very simple—we can dedicate a forwarding engine to each virtual network and ensure that packets never cross (and risk confusion) in the forwarding path.

Placing forwarding engines on separate cards led to a fourth innovation. Because the forwarding engines are separate from the line cards, they may receive packets from line cards that are not destined to them. The line card must then forward these packets to the correct forwarding engine, which can add significant overhead to the processing of these packets. To address this issue, we developed a common intermediate format across protocols that allows the forwarding engine to process packets without the need for the line card to perform additional processing.

Full routing table local to each FE

Switched backplane

Common intermediate format across protocols

Not shown: Network processor to handle special case packets

[Partridge et al ’98]
Switching fabric

Operates in *epochs*

- **128 bytes sent by each line card to next-hop line card**
- **Each line card can send to only one other card, and can receive from only one other card**

... to outputs

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Inputs ready to send...

Allocator assigns inputs to outputs & tells line cards

What fundamental problem is being solved?
Maximum bipartite matching

Maximize number of matched input-output pairs, such that each input & output only matched once

50 Gbit/s router uses approximate solution

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Inputs ready to send...
Many functions in modern routers

- access control
- quality of service
- accounting, traffic metering
- IPv4, IPv6, MPLS, ethernet, ...
- Virtual private networks
- ...
### Efficiency vs. extensibility

<table>
<thead>
<tr>
<th>Hardware routers</th>
<th>Software routers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast</td>
<td>Slow</td>
</tr>
<tr>
<td>Specific functionality</td>
<td>Extensible</td>
</tr>
<tr>
<td>Result: many physical</td>
<td></td>
</tr>
<tr>
<td>devices (routers,</td>
<td></td>
</tr>
<tr>
<td>firewalls, intrusion</td>
<td></td>
</tr>
<tr>
<td>detection, ...)</td>
<td></td>
</tr>
</tbody>
</table>

Can we get the best of both worlds?
RouteBricks approach

- **Parallelism within servers**
- **Parallelism across servers**
- **High bandwidth switching fabric built from commodity hardware**

[Image: [Dobrescu et al, NSDI 2009]]
Switching fabric challenges

Handle any traffic pattern: for example, all input traffic at a server might go to any one output server.

Low degree: we’re using commodity hardware.

Naïve approach:

Useless: might as well just use one server!
Low degree solution

Just one link out for each link in

Total out b/w enough, but doesn’t go where we need

Solution (**Valiant load balancing**): send packet to random intermediate node, then on to destination
Guaranteed to nearly full throughput for any traffic demands!

- “nearly” = 2x. Why?
- So, switch fabric needs to be 2x as fast as external links to provide guarantees

Why does sending to a random intermediate node work?

Still using one port per server. What if # servers > # ports available?
MPLS design

Ingress:
Traffic classification, label packets ("forwarding equivalence class")

Control plane constructs label-switched paths and coordinates labels

Can also stack labels = concatenate paths
MPLS motivation

In the design doc

- High performance forwarding
- Minimal forwarding requirements, so can interface well with many types of media such as ATM
- Flexible control of traffic routing

What matters today? **Flexibility**. Widely used for:

- Virtual Private Network (VPN) service along dedicated paths between enterprise sites
- Traffic engineering on per-“flow” granularity
- Control backup paths with MPLS Fast ReRoute
Announcements

By tonight:

• Submit project proposal
• Fill out presentation topic survey

Thursday:

• BGP routing policies in ISP networks (Caesar and Rexford, 2005)
• Class ends 4:30 p.m.