Fast Reroute and Multipath Routing Extensions to the NetFPGA Reference Router

Hongyi Zeng, Mario Flajslik, Nikhil Handigol
Department of Electrical Engineering and Department of Computer Science
Stanford University
Stanford, CA, USA
{hyzeng, mariof, nikhilh}@stanford.edu

ABSTRACT

This paper describes the design and implementation of two feature extensions to the NetFPGA reference router - fast reroute and multipath routing. We also share our insight into the inherent similarities of these two seemingly disparate features that enable us to run them simultaneously. Both features are designed to work at line-rate. With minimum modification of both hardware and software, the advanced features are tested and will be demonstrated on the NetFPGA card.

1. INTRODUCTION

One of the many systems built using NetFPGA is an IPv4 reference router [1]. The router runs the Pee-Wee OSPF [2] routing protocol, and does address lookup and packet forwarding at line-rate.

In this paper, we present two feature extensions to the NetFPGA reference router:

- *Fast reroute* - Detection of link failure or topology change in the reference router is generally based on the OSPF messages timing out. However, this causes packets to be dropped in the interval between the actual failure and failure detection. These intervals are as large as 90 seconds in PW-OSPF. Fast reroute [3] is a technique that detects link failures at the hardware level and routes packets over alternative routes to minimize packet drops. These alternative routes are pre-computed by the router software.
- *Multipath routing* - Multipath routing [4] is a routing strategy where next-hop packet forwarding to a single destination can occur over multiple "best paths". This enables load-balancing and better utilization of available network capacity. Our implementation of multipath routing is similar to ECMP; packets are forwarded over only those paths that tie for top place in routing metric calculations. This has the two-fold advantage of keeping the routing protocol simple and robust as well as minimizing packet reordering.

This work was originally intended as an advanced feature project for CS344 "Building an Internet Router" class, in the year 2009 at Stanford University.

2. DESIGN

The main goal of CS344 class is to design an output port lookup module for the NetFPGA. This module takes incoming packets, parses header information, queries the routing table and ARP cache, labels the packet with output port information, and finally puts it in output queues. Along with other modules in NetFPGA gateware, a functional Internet router can be built.

2.1 Architecture

The overall architecture of output port lookup module is shown in Figure 1.

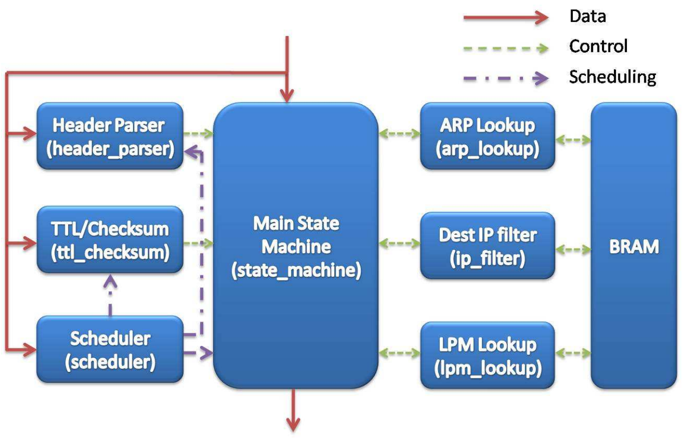
Data
Control
Scheduling
Header Parser (header_parser)
TTL/Checksum (ttl_checksum)
Scheduler (scheduler)
Main State Machine (state_machine)
ARP Lookup (arp_lookup)
Dest IP filter (ip_filter)
LPM Lookup (lpm_lookup)
BRAM

Figure 1: Block Diagram of Output Port Lookup

The scheduler provides "position" information to other modules. This simplifies the design of header parser and TTL/Checksum. The header parser parses the header of packets, and TTL/Checksum module manipulates TTL/Checksum information of IP packets.

There are three table lookup modules for ARP table, IP filter table, and routing table. The first two have similar lookup mechanism, while routing table lookup should be Longest Prefix Matching (LPM). In general, these modules accept a search key and a REQ signal, feed back an ACK signal with the results. On the other side, these modules connect to the Block RAM (BRAM) interface provided by Xilinx. Table entries are stored in BRAM.

The main state machine reads in the entire header to a FIFO. At the same time header_parser and ttl_checksum prepares the necessary information to the state machine. If the packet is a regular IP packet, the state machine issues a IP filter search request. If the address is found in the IP filter table, the packet will be kicked up to software. Otherwise, the state machine does a routing table search, and a ARP search. In the last stage, the state machine modifies the MAC header, TTL, and checksum, and sends the packet to the destination port.

The extension code to support fast reroute and multipath routing is mainly in the routing table and lpm_lookup module. We will describe the two new features in the following subsections. Before that, the routing table structure and LPM lookup process will be presented.

2.2 Routing Table and LPM Lookup

2.2.1 Routing Table

Each entry of the routing table consists four parts: IP address as search key, the mask, next-hop IP, and port. The port information is stored as a one-hot-encoded number. This number has a one for every port the packet should go out on where bit 0 is MAC0, bit 1 is CPU0, bit 2 is MAC1, etc. The structure of the entry is depicted in Table 1.

Search IP	Mask	Next-hop IP	Port
192.168.100.0	255.255.255.0	192.168.101.2	0000 0001

Table 1: Entry Structure of the Routing Table

2.2.2 LPM Lookup

Due to the course requirement, we did not use the Xilinx Ternary Content Addressable Memory (TCAM) cores[5, 6]. Instead, we implement the routing table with BRAM on the NetFPGA card. Linear search is employed in LPM lookup as the size of the routing table is relatively small (32 entries required by the class). The entries with longer prefix are stored in front of those with shorter prefix. By doing this, entries with longest prefix will naturally come out first in a linear search.

2.3 Fast Reroute

In order to realize fast reroute feature, the router software needs to store a backup path for those "fast reroute protected" entry. In our router, the backup path information is in the form of duplicate entries only with *different port information*. In the normal case, the lpm_lookup module will return the *first* matched entry to the main state machine, making the port in this entry having the highest priority. When the port in the primary entry fails, the second entry with backup port information will be used and the flow will be rerouted. Table 2 is an example.

	Search IP	Mask	Next-hop IP	Port
1	192.168.100.0	255.255.255.0	192.168.101.2	0000 0001
2	192.168.100.0	255.255.255.0	192.168.101.2	0000 0100

Table 2: Fast Reroute entries. The primary port is MAC0. The backup port is MAC1

The reroute procedure is very fast because it is purely based on hardware. We make use of in-band link status information from Broadcom PHY chips as feedback. Once a link is down, lpm_lookup module will notice this immediately. The next coming packet will not follow the entry with invalid output port. Further details on in-band status information of NetFPGA's PHY chip can be found in Broadcom's BCM5464SR data sheet[7].

Besides link status feedback, the router hardware needs no modification under a linear search scheme. However, the duplicate entries will take up extra space in the routing table. At the same time, it is not applicable to TCAM based lookup mechanism, in which entries are not stored in order. Our solution is to extend port information section in the entry from 8bit to 16bit. The first 8bit is the primary port while the following 8bit is the backup. The primary port will be used first unless the associated link is down.

2.4 Multipath Routing

In the NetFPGA reference router, a routing table entry with multiple 1's in port section indicates itself as a multicast entry. Packets match this entry are sent to those ports at the same time. Based on the fact that in the current OSPF routing protocol, a packet is never sent to more than one port, we decided to take advantage of this section to implement multipath routing.

The goal of multipath routing is to allow packets destined to the same end-host making use of more than one route. In our multipath routing implementation, each entry in the routing table may have more than one output port, with multiple 1's in port section. Packets matching this entry could go to any port indicated in the entry. Note that for multipath entry, each output port will have its correspondent next-hop IP (gateway). We created another gateway table to store the gateway address. In routing table, we store a 8bit pointer (index) for each output port that can be used. Currently we use a simple round-robin fashion to choose the actual output port. A register keeps track of which port was last used and instructs lpm_lookup module to find the next available port. A multipath entry and gateway table example is shown in Table 3.

Search IP	Mask	Next-hop IP index	Port
192.168.100.0	255.255.255.0	02 00 00 01	0100 0001

Index	Next-hop IP
1	192.168.101.2
2	192.168.102.3

Table 3: Multipath entry. Packets use MAC0, MAC3 in turns.

We do not specify the priority of ports in the same entry. Each port, if available, will be used with equal probability. However, priority can still be realized by ordered duplicate entries described in the last section. One may optimize bandwidth, delay, quality of service, etc. by choosing the output port cleverly.

It is worth to point out that, unlike fast reroute, the multipath routing implementation is independent of how entries are stored. The same code applies to TCAM based router.

2.5 Limitation

We understand that there are a number of limitation in the design.

First, for fast reroute feature, the only feedback information is the link status. However, when the neighbor router goes down or freezes, sometimes the link status may remain active. In this case, it will not trigger the fast rerouting mechanism, and the application is subject to interruption. By design, our implementation is a hardware based improvement to the current OSPF protocol. With the software, the topology is still recalculated regularly to overcome the router failures not resulting an inactive link state.

Another limitation of the design is packet reordering. We split a single flow into multiple paths without packet reordering protection. Packets could arrive at the destination in different order as they are sent. As the hardware router providing an interface to handle multipath routing, the software (multipath routing protocol, transport layer protocol such as TCP, or applications) may develop some methods to ensure the quality of service.

3. IMPLEMENTATION

3.1 Hardware

Fast reroute and multipath routing features have already been implemented in the hardware with linear search based implementation. The corresponding Verilog code is less than 100 lines.

In general, the two advanced features consume little logics in FPGA. However, duplicate entries for fast reroute may need more BRAMs to store. Table 4 describes the device utilization of out project. It uses 31% of the available flip-flops on the Xilinx Virtex II Pro 50 FPGA, which is almost equal to the reference router. 50% of the BRAMs available are used. The main use of BRAMs occurs in the three tables.

Resources	XC2VP50 Utilization	Utilization Percentage
Occupied Slices	14,781 out of 23,616	62%
4-input LUTS	17,469 out of 47,232	36%
Flip Flops	14,918 out of 47,232	31%
Block RAMs	118 out of 232	50%
External IOBs	356 out of 692	51%

Table 4: Device utilization for Fast Reroute and Multipath Routing enabled Router

3.2 Software

Software is responsible for providing correct tables to the hardware. Fast reroute and multipath routing features require changes only to the routing table. In the basic router implementation the Routing Table is generated using Dijkstra's algorithm to find shortest path to all known destinations. This calculation is done whenever the topology changes, as perceived by the router in question. To support fast reroute and multipath routing, it is not sufficient to find shortest paths to all destinations, but also second shortest (and possibly third shortest and more) paths are also necessary. This is calculated by running Dijkstra's algorithm four times (because NetFPGA has four interfaces). For each run of the algorithm, all interfaces on the router are disabled, except for one (a different interface is enabled in each run). Resulting hop count distances (i.e. the distance vectors) for each of the algorithm runs are then compared to provide the routing table.

3.2.1 *Fast Reroute*

When fast reroute feature is enabled, two entries for each destination will be added to the routing table if the destination can be reached over at least two interfaces. The first entry corresponds to the shortest path route and is preferred, and the second entry is the backup path if the primary path is disabled. The router will be in the mode where it uses a backup path only for the short time that it will take OSPF to update all routers. After that, the backup path will become the primary path, and a new backup path will be calculated (if available). Because of this, adding a second backup path, while possible, is deemed unnecessary. Also, this mechanism allows fast reroute to be enabled for some chosen routes, and disabled for others, which can potentially save space in the routing table.

3.2.2 *Multipath Routing*

Equal cost multipath has been chosen for its simplicity in implementation and limited packet reordering. To implement this we search if each of the destinations can be reached over multiple interfaces (as calculated by different algorithm runs) in the same minimum hop count. If this is so, all such interfaces are added to the routing table entry, if not, only the shortest path interface is added to the routing table.

In order to measure performance and demonstrate how fast reroute and multipath routing work, a demo application is being developed. This application consists of a GUI and a backend. The backend communicates with all routers and collects statistical information, such as packet count for each interface of each router. It is also aware of the network topology, which it then feeds to the GUI for visual presentation, together with the statistical data. Results will be available soon, as the demo application is completed.

4. CONCLUSION

In this paper we described the design and implementation of the fast reroute and multipath routing extensions to the NetFPGA reference router. Implemented with very little modification to the hardware pipeline, these features enhance the robustness and efficiency of the network. In addition, the GUI frontend can be used to visualize and validate the performance of the system.

This work is based on a beta version of the NetFPGA gateware, which lacks TCAM cores and SCONE (Software Component Of NetFPGA). In the future, we will port the code to NetFPGA beta-plus version, in order to achieve higher performance and reliability.

5. REFERENCES

[1] NetFPGA Group, "NetFPGA reference router," http://netfpga.org/wordpress/netfpga-ipv4-reference-router/.

[2] Stanford University CS344 Class, "Pee-Wee OSPF Protocol Details," http://yuba.stanford.edu/cs344/pwospf/.

[3] P. Pan, G. Swallow, and A. Atlas, "Fast Reroute Extensions to RSVP-TE for LSP Tunnels," RFC 4090 (Proposed Standard), Internet Engineering Task Force, May 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4090.txt

[4] D. Thaler and C. Hopps, "Multipath Issues in Unicast and Multicast Next-Hop Selection," RFC 2991 (Informational), Internet Engineering Task Force, Nov. 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2991.txt

[5] Xilinx, Inc, "An overview of multiple CAM designs in

Virtex family devices," http://www.xilinx.com/support/documentation/application_notes/xapp201.pdf.

[6] ——, "Designing flexible, fast CAMs with Virtex family FPGAs," http://www.xilinx.com/support/documentation/application_notes/xapp203.pdf.

[7] Broadcom Corporation, "BCM5464SR Quad-Port 10/100/1000BASE-T Gb Transceiver with Copper/Fiber Media Interface," http://www.broadcom.com/products/Enterprise-Networking/Gigabit-Ethernet-Transceivers/BCM5464SR.