Lab 9 - Custom FTP on Custom Network
Group: Route-On

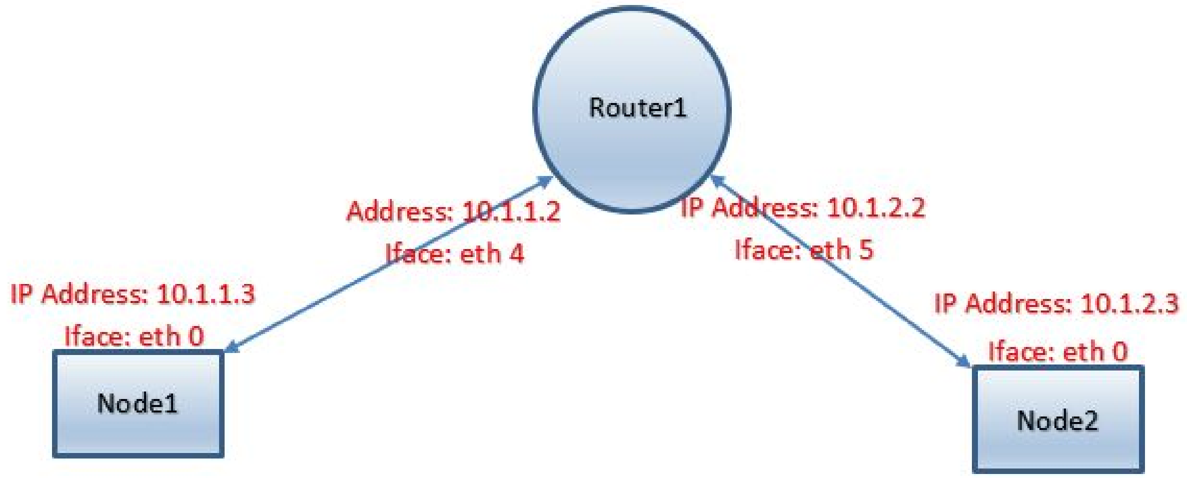
Group Member:
Hao Zhang, Vidhi Goel, Venkata Srinivas Chowdary Reddy

Oct 17th, 2015

A simple Custom Packet and Custom Protocol was built in our previous lab. Now we are going to use those to do a Custom File Transfer implementation on this Custom Network. For this lab we are modifying our Custom FTP used in Lab 5.

DESIGN IMPLEMENTATION

For testing our Custom FTP we are considering a simple Node - Router - Node topology as shown below:



We are making a file transfer from Node1 to Node2 through Router1.

<u>FTP program with Custom Protocol</u>

With our Custom Protocol we have two operational modes. Reliable transfer & Unreliable transfer. For this lab, we would demonstrate the Reliable File transfer where we track incoming packets using their sequence numbers and write the packet payload to the file. For missing sequence packets we send NACK's to the sender. In response to NACK's the sender retransmits those packets. In Unreliable File transfer, the receiver write the packet payload to the file and there is no tracking of incoming packets.

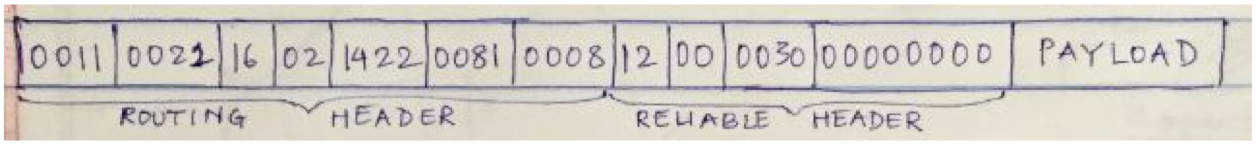
Our custom protocol for reliable transfer looks like below:

```
/* Routing protocol header */
        typedef struct rthdr {
                u_int16_t       saddr;          /* source address */
                u_int16_t       daddr;          /* destination address */
                u_int8_t        ttl;            /* time-to-live: for killing unintended long path, which might
lead to congestion */
                u_int8_t        protocol;       /* route-on team defined protocol */
                u_int16_t       size;           /* packet length */
                u_int16_t       check;          /* routing protocol header checksum */
                unsigned char   dummy[4];
        }  rthdr_t;

/* reliable transfer protocol */
        typedef struct rlhdr {
                u_int8_t        port;           /* application instance port */
                u_int8_t        dummy;
                u_int16_t       check;          /* reliable transfer protocol header + data checksum */
                u_int32_t       seq;            /* transfer sequence number */

        } rlhdr_t;
```

FILE TRANSFER Packet



Above is the file transfer packet that contains the actual PAYLOAD sent from Sender to Receiver. For ease of understanding, some of the fields are shown in decimal although we use hexadecimal format in our program.

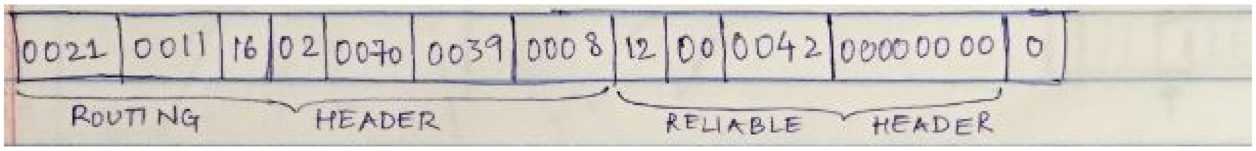
Routing header is briefly described as below:
Source - 11,
Destination - 21,
TTL - 16,
Protocol - 02 (Reliable File transfer protocol).
Size - 1422 (Total size including headers),
Checksum - 0081 (This is checksum for routing protocol header alone. Router will verify this checksum)
Dummy - 0008 (This is used to simulate the protocol field of ethernet header. NIC card rejects the packet if this is not set correctly).

Reliable File transfer protocol header is described as below:
Port - 12 (This is the application port number)

Dummy - 00 (This field is optional and can be used for additional purposes)
Checksum - 0030 (This is checksum for reliable header and application payload. End system needs to verify this)
Sequence Number - 00000000 (This is the first sequence number sent by Sender to Receiver)

PAYLOAD correspond to the actual file data sent from Sender to Receiver

NACK Packet



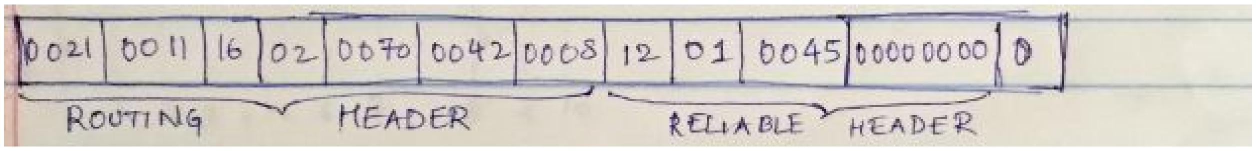
0021	0011	16	02	0070	0039	0008	12	00	0042	00000000	0
ROUTING			HEADER				RELIABLE		HEADER		

This is sample of a NACK packet sent from Receiver to Sender in order to indicate that a particular sequence number has not been received.

All the fields are self-explanatory. Some of the fields that need to be described are as follows:
Size - 70 (FIxed the NACK packet size to be 70, as 60 is the minimum size that we can inject from NIC)
Sequence Number - 00000000 (This will correspond to the sequence number missing at receiver)
***Payload - 0** (All the remaining bytes of the packet besides the header are set to 0.)*

The Sender recognizes the packet to be a NACK by checking if the PAYLOAD is 0.

END SESSION Packet



0021	0011	16	02	0070	0042	0008	12	01	0045	00000000	0
ROUTING			HEADER				RELIABLE		HEADER		

When Receiver receives the entire file, it sends a END SESSION packet to the Sender to ensure that Sender reads this packet and terminates its execution.
The changed fields in this case are as below:

Size - 70 (To send a small packet that can go through the NIC, we use the minimum safest size)
***Dummy - 01** (Dummy field of Reliable Header Protocol)*

We are utilizing the dummy field of Reliable Header Protocol to notify the END SESSION event to the Sender. The Sender will check if the Dummy field is set to 1, it will mark it as End of session. Else, it will continue to process NACK and retransmit lost packets.

CODE REVIEW

Packet.h

- Define the structure for routing protocol header
- Defines the structure for application header - Control protocol header, unreliable transfer protocol and reliable transfer protocol

sender.c

- We record a timestamp when we start the program
- We first send all file segments once
- Then we wait for nacks from the receiver
- If we receive a nack, we retransmit the corresponding file segment
- The receiver sends an END using the dummy field of Reliable transfer protocol header, when all the files segments are successfully received, and once we receive the END, we record another timestamp
- We calculate the transmission time and the throughput
- We did the following optimizations
    - Map the file to memory when reading it
    - Dedicated iface handler for sending and receiving. So sending file segments and receiving nacks can be done in parallel

receiver.c

- Receiver runs the process packet in main thread to receive the packets from the router.
- It maintains a Tracking Window (array) to monitor which packets have been received.
- When a new packet is received, it checks if the packet is already received or not. If yes, do nothing.
- Else, if it is a new packet, it updates the tracking window, writes the payload to the file and continue listening.
- Meanwhile, a thread is started to send the NACK to sender for the lost packets.
- The thread generates NACK packet and inject it into the interface which is then forwarded by the router.
- Receiver continue to run until all the packets are received. It then sends a END packet to Sender using the dummy field of Reliable Header and terminates.

router.c

- Creates routing table
- Sniff all available device and filter out valid network interfaces
- Assign thread to each device
- Each thread create private handle for all devices, and initialize its own stat data
- All threads start sniffing and process packet in parallel

    - We first verify routing header checksum, if it does not pass (it could be some data communication with control network), we simply drop it
    - If the packet needs forwarding, the thread modifies the packet and inject it to the target interface
- Modification / Optimization we made
    - Each sniff thread has private handles for each interface, as pcap is not thread safe
    - When routing, the router only checks the checksum of the routing header, which is time saving. The end user will check the checksum for reliable header and payload
    - Use as many memory operation as possible
    - Re-use constant buffers, no dynamic allocation, which saves time
    - Routing table is stored in u_int64_t, and we use bit manipulation for routing lookup, which means we perform the entire lookup within CPU registers, and therefore we saves the time even for accessing cache. (This optimization might not be obvious when the routing table is small.)

packet_util.c

- Checksum generation and its verification
- Packet generation and its verification

printp.c

- The famous route-on fprintp() function sits here
- It has customized print option based on packet type
- It displays meaningful information in the packet, and the raw data as well. The output format follows the one used by wireshark

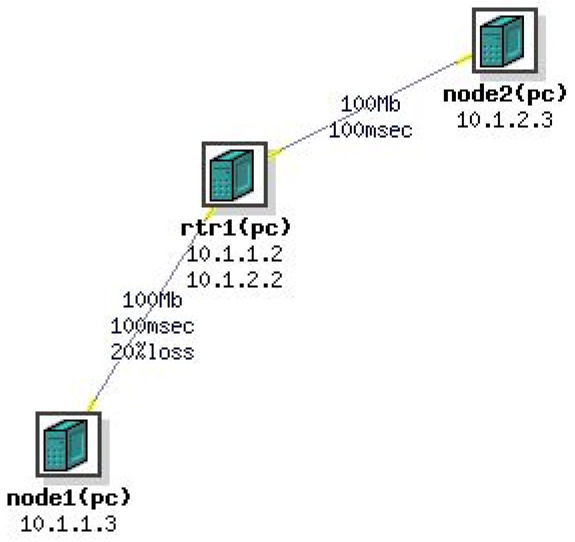
DETER IMPLEMENTATION

NS File:

```
set ns [new Simulator]
source tb_compat.tcl
# Nodes
set rtr1 [$ns node]
tb-set-node-os $rtr1 Ubuntu1004-STD
set node1 [$ns node]
tb-set-node-os $node1 Ubuntu1004-STD
set node2 [$ns node]
tb-set-node-os $node2 Ubuntu1004-STD
set link0 [$ns duplex-link $rtr1 $node1 100Mb 100ms DropTail]
```

```
tb-set-link-loss $link0 0.20
set link1 [$ns duplex-link $rtr1 $node2 100Mb 100ms DropTail]
$ns rtproto Manual
$ns run
```

Visualization:



Measuring "Packets Per Second" (PPS):

Packets Per Second is one of the performance metric for Network Devices such as Router and Switches. The Packet Forwarding Rate (PPS) depends on the packet size. At smallest packet size one can achieve Maximum PPS. Thus when we need Max. PPS we need to forward smallest possible packets through router.

Packets Per Second = Throughput (Byte/sec) / Packet size (Byte)

TEST RESULTS

We tested the router's RAW packet process rate (the sender keeps on sending packets, without any link loss), and the average RAW packet process rate, single way, is **94,516,249.47** bits/second, on a 100Mbps link.

File Size (Byte)	Segment Size (Byte)	Packet Size (Byte)	File Transfer Throughput (Bit/sec)	Router Single Way Process Rate (Bit/sec, 20% link loss)	Router Single Way Packet Rate (PPS)
1048576000	1400	1422	63494428.78	76097822.43	6690.68
524288000	1400	1422	61741889.88	77922714.69	6848.75

209715200	1400	1422	62015784.21	80054260.57	7073.16
104857600	1400	1422	59268680.46	79032458.94	6947.30

Sample Script file we used for throughput calculation:

```
#!/bin/sh
echo "===>start timestamp:"
A=$(date +%s%N)
echo $A
sudo ./sender data1000.bin node4 43
B=$(date +%s%N)
echo "===>end timestamp:"
echo $B
echo "===>Throughput"
echo "1048576000 / ($B - $A) * 1000000000 * 8" | bc -l
echo "bits/second"
```

100MB File:

Average File Transfer Rate: 59,268,680.46 bits/sec

Average Router Single Direction Process Rate: 79,032,458.94 bits/sec (20% link loss)

Sample Sender Output 1

```
sc558cm@node3:/tmp/lab9/FT$ ./sender.sh
===>start timestamp:
1445113867986525660
Find iface eth3, ip 10.1.1.3
Trying to open device eth3 to send ... OPEN DONE
Filesize is 104857600

TOTAL PACKETS TRANSMITTED: 74899
TOTAL EXECUTION TIME 13.077033
===>end timestamp:
1445113881730306927
===>Throughput
61035662.87206395480000000000
bits/second
```

Sample Router Stat 1

```
thread 0: received a P_FORWARD packet of size 1422, inject to iface[3]; 94038 packets
(133720036 bytes) processed
```

Sample Sender Output 2

```
sc558cm@node3:/tmp/lab9/FT$ ./sender.sh
===>start timestamp:
1445114150334519664
Find iface eth3, ip 10.1.1.3
```

```
Trying to open device eth3 to send ... OPEN DONE
Filesize is 104857600

TOTAL PACKETS TRANSMITTED: 74899
TOTAL EXECUTION TIME 13.735940
===>end timestamp:
1445114164742329070
===>Throughput
58222646.92442864480000000000
bits/second
```

Sample Router Stat 2

```
thread 0: received a P_FORWARD packet of size 1422, inject to iface[3]; 92024 packets
(130855128 bytes) processed
```

Sample Sender Output 3

```
sc558cm@node3:/tmp/lab9/FT$ ./sender.sh
===>start timestamp:
1445114665654519324
Find iface eth3, ip 10.1.1.3
Trying to open device eth3 to send ... OPEN DONE
Filesize is 104857600

TOTAL PACKETS TRANSMITTED: 74899
TOTAL EXECUTION TIME 13.635525
===>end timestamp:
1445114679982329763
===>Throughput
58547731.60012212800000000000
bits/second
```

Sample Router Stat 3

```
thread 0: received a P_FORWARD packet of size 1422, inject to iface[3]; 94950 packets
(135016900 bytes) processed
```

200MB File:

Average File Transfer Rate: 62,015,784.21 bits/sec
Average Router Single Direction Process Rate: 80,054,260.57 bits/sec (20% link loss)

Sample Sender Output 1

```
sc558cm@node3:/tmp/lab9/FT$ ./sender.sh
===>start timestamp:
1445111993026601559
```

```
Find iface eth3, ip 10.1.1.3
Trying to open device eth3 to send ... OPEN DONE
Filesize is 209715200

TOTAL PACKETS TRANSMITTED: 149797
TOTAL EXECUTION TIME 25.903545
===>end timestamp:
1445112019614352137
===>Throughput
63101299.0391232486400000000
bits/second
```

Sample Router Stat 1

```
thread 0: received a P_FORWARD packet of size 1422, inject to iface[3]; 187094 packets
(266045868 bytes) processed
```

Sample Sender Output 2

```
sc558cm@node3:/tmp/lab9/FT$ ./sender.sh
===>start timestamp:
1445112215262590076
Find iface eth3, ip 10.1.1.3
Trying to open device eth3 to send ... OPEN DONE
Filesize is 209715200

TOTAL PACKETS TRANSMITTED: 149797
TOTAL EXECUTION TIME 27.097735
===>end timestamp:
1445112243022336092
===>Throughput
60437210.0174477331200000000
bits/second
```

Sample Router Stat 2

```
thread 0: received a P_FORWARD packet of size 1422, inject to iface[3]; 180394 packets
(256519068 bytes) processed
```

Sample Sender Output 3

```
sc558cm@node3:/tmp/lab9/FT$ ./sender.sh
===>start timestamp:
1445112627126587166
Find iface eth3, ip 10.1.1.3
Trying to open device eth3 to send ... OPEN DONE
Filesize is 209715200

TOTAL PACKETS TRANSMITTED: 149797
TOTAL EXECUTION TIME 26.176209
```

```
===>end timestamp:
1445112653966335031
===>Throughput
62508843.54199949560000000000
bits/second
```

Sample Router Stat 3

```
thread 0: received a P_FORWARD packet of size 1422, inject to iface[3]; 189714 packets
(269772108 bytes) processed
```

500MB File:

File segsize = 1400 Bytes.

Average File Transfer Rate: 61,741,889.88 bits/sec

Average Router Single Direction Process Rate: 77,922,714.69 bits/sec (20% link loss)

Sample Sender Output 1

```
sc558cm@node3:/tmp/lab9/FT$ ./sender.sh
===>start timestamp:
1445110092502603925
Find iface eth3, ip 10.1.1.3
Trying to open device eth3 to send ... OPEN DONE
Filesize is 524288000

TOTAL PACKETS TRANSMITTED: 374492
TOTAL EXECUTION TIME 64.756367
===>end timestamp:
1445110157926391074
===>Throughput
64109770.81543207128000000000
bits/second
```

Sample Router Stat 1

```
thread 0: received a P_FORWARD packet of size 1422, inject to iface[3]; 448775 packets
(638157250 bytes) processed
```

Sample Sender Output 2

```
sc558cm@node3:/tmp/lab9/FT$ ./sender.sh
===>start timestamp:
1445110555998586891
Find iface eth3, ip 10.1.1.3
Trying to open device eth3 to send ... OPEN DONE
Filesize is 524288000

TOTAL PACKETS TRANSMITTED: 374492
TOTAL EXECUTION TIME 69.956596
===>end timestamp:
```

```
1445110626650405222
===>Throughput
59365832.31800078376000000000
bits/second
```

Sample Router Stat 2

```
thread 0: received a P_FORWARD packet of size 1422, inject to iface[3]; 489612 packets
(696226664 bytes) processed
```

Sample Sender Output 3

```
sc558cm@node3:/tmp/lab9/FT$ ./sender.sh
===>start timestamp:
1445110930302510917
Find iface eth3, ip 10.1.1.3
Trying to open device eth3 to send ... OPEN DONE
Filesize is 524288000

TOTAL PACKETS TRANSMITTED: 374492
TOTAL EXECUTION TIME 67.245565
===>end timestamp:
1445110998226389165
===>Throughput
61750066.51837492992000000000
bits/second
```

Sample Router Stat 3

```
thread 0: received a P_FORWARD packet of size 1422, inject to iface[3]; 444981 packets
(632761382 bytes) processed
```

1GB File:

File segsize = 1400 Bytes.

Average File Transfer Rate: 63,494,428.7 bits/sec

Average Router Single Direction Process Rate: 76,097,822.43 bits/sec (20% link loss)

Sample Sender Output 1

```
sc558cm@node3:/tmp/lab9/FT$ ./sender.sh
===>start timestamp:
1445106069458495702
Find iface eth3, ip 10.1.1.3
Trying to open device eth3 to send ... OPEN DONE
Filesize is 1048576000

TOTAL PACKETS TRANSMITTED: 748983
TOTAL EXECUTION TIME 124.758701
===>end timestamp:
```

```
1445106194898448296
===>Throughput
66873494.66043437392000000000
bits/second
```

Sample Router Stat 1

```
thread 0: received a P_FORWARD packet of size 1422, inject to iface[3]; 898080 packets
(1277069360 bytes) processed
```

Sample Sender Output 2

```
sc558cm@node3:/tmp/lab9/FT$ ./sender.sh
===>start timestamp:
1445106916486618937
Find iface eth3, ip 10.1.1.3
Trying to open device eth3 to send ... OPEN DONE
Filesize is 1048576000

TOTAL PACKETS TRANSMITTED: 748983
TOTAL EXECUTION TIME 140.133777
===>end timestamp:
1445107057318411614
===>Throughput
59564732.08602412992000000000
bits/second
```

Sample Router Stat 2

```
thread 0: received a P_FORWARD packet of size 1422, inject to iface[3]; 841689 packets
(1196881358 bytes) processed
```

Sample Sender Output 3

```
sc558cm@node3:/tmp/lab9/FT$ ./sender.sh
===>start timestamp:
1445107757914644146
Find iface eth3, ip 10.1.1.3
Trying to open device eth3 to send ... OPEN DONE
Filesize is 1048576000

TOTAL PACKETS TRANSMITTED: 748983
TOTAL EXECUTION TIME 130.281517
===>end timestamp:
1445107888894425873
===>Throughput
64045060.15656906056000000000
bits/second
```

Sample Router Stat 3

```
thread 0: received a P_FORWARD packet of size 1422, inject to iface[3]; 904214 packets
(1285791508 bytes) processed
```

No Loss Router Single Way Process Rate Test:

Average Router Single Direction Process Rate: 94,516,249.47 bits/sec (0% link loss)

Sample Sender Output 1

```
sc558cm@node3:/tmp/lab9/FT$ ./sender.sh
===>start timestamp:
1445116248822511341
Find iface eth3, ip 10.1.1.3
Trying to open device eth3 to send ... OPEN DONE
Filesize is 1048576000
TOTAL PACKETS TRANSMITTED: 748983
TOTAL EXECUTION TIME 111.355489
===>end timestamp:
1445116360866936126
===>Throughput
74868589.09844685848000000000
bits/second
```

Sample Router Stat 1

```
thread 0: received a P_FORWARD packet of size 1422, inject to iface[3]; 945419 packets
(1344385618 bytes) processed
```

Sample Sender Output 2

```
sc558cm@node3:/tmp/lab9/FT$ ./sender.sh
===>start timestamp:
1445117124870504472
Find iface eth3, ip 10.1.1.3
Trying to open device eth3 to send ... OPEN DONE
Filesize is 1048576000
TOTAL PACKETS TRANSMITTED: 748983
TOTAL EXECUTION TIME 109.270398
===>end timestamp:
1445117234874454194
===>Throughput
76257334.58843558808000000000
bits/second
```

Sample Router Stat 2

```
thread 0: received a P_FORWARD packet of size 1422, inject to iface[3]; 908789 packets
(1292297758 bytes) processed
```

Sample Sender Output 3

```
sc558cm@node3:/tmp/lab9/FT$ ./sender.sh
===>start timestamp:
1445119240330474755
Find iface eth3, ip 10.1.1.3
Trying to open device eth3 to send ... OPEN DONE
```

```
Filesize is 1048576000

TOTAL PACKETS TRANSMITTED: 748983
TOTAL EXECUTION TIME 112.085881
===>end timestamp:
1445119353117462796
===>Throughput
74375671.74815057080000000000
bits/second
```

Sample Router Stat 3

```
thread 0: received a P_FORWARD packet of size 1422, inject to iface[3]; 910077 packets
(1294129294 bytes) processed
```