

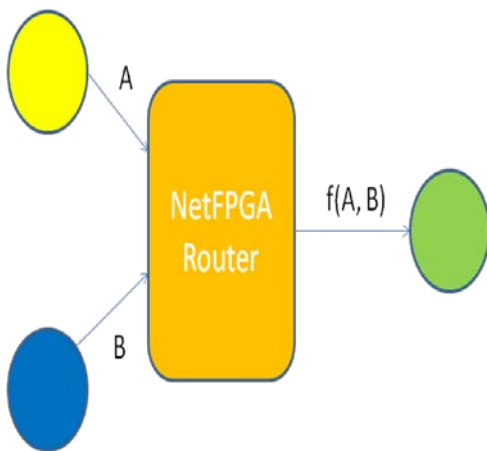
GEC9 Demo : Network Coding in NetFPGA Routers (Mobile Gigabit Wireless Access)

Participants: [Parameswaran Ramanathan, University of Wisconsin, Madison](#)
[Kaung-Ching Wang, Clemson University](#)

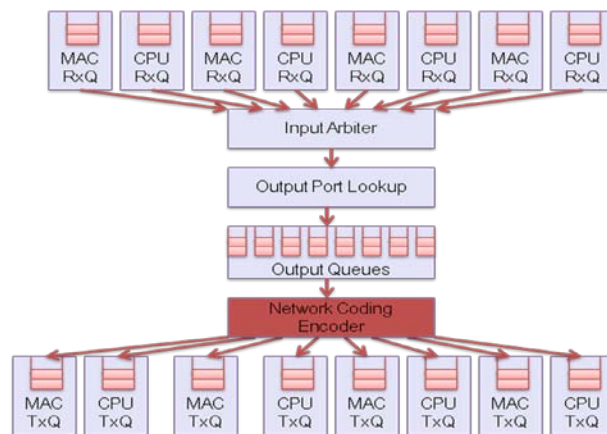
URL: <http://www.ece.wisc.edu/~parmesh/geni.html>

Abstract

Emerging gigabit wireless technologies will soon allow mobile devices to access broadband media on the move via gigabit wireless networks on the edge of the future Internet. However, gigabit wireless access does not easily result in gigabit end-to-end throughputs to mobile devices. The main goal of this project is to experiment with techniques needed to support end-to-end gigabit TCP and UDP throughputs from the Internet to the mobile devices. Specifically, the project proposes to leverage GENI's deeply programmable core and wireless edge networks, to experiment with techniques such as inter-flow and intra-flow network coding, multipath routing, and on-demand localized congestion control schemes in OpenFlow routers. Ongoing research shows that these components are key parts of the solution needed to deliver gigabit end-to-end throughputs to mobile users.



(a) Host configuration



(b) Block diagram of NetFPGA router



The demonstration here shows one of the first steps towards realizing the above experimental plan. In particular, it demonstrates inter-flow network coding in a NetFPGA router. As shown in the figure above, two hosts, namely Host 1 and Host 2, send a stream of UDP packets to a common receiver in Host 3 through a NetFPGA router. Instead of forwarding the packets as received, the NetFPGA router linearly combines a packet from Host 1 with a packet from Host 2 and forwards this network coded packet to Host 3. To perform this network coding in hardware, we have modified the design of a reference router distributed by the team at Stanford University. Specifically, as shown in the figure above, we added a new functional block called the *Network Coding Encoder* in the packet processing pipeline of the router.

In the demonstration, an observer can specify an 8-byte data payload to be included in each UDP/IP packet from Host 1. Similarly, one can also specify an 8-byte data payload to be included in each UDP/IP from Host 2. On the projected screen, one can observe the data payload in the Ethernet frames forwarded to Host 3 by the NetFPGA router. Since these Ethernet frames encapsulate the network coded UDP/IP packets, the data payload will contain a pre-determined linear combination of the observer specified 8-byte data payloads included in the UDP/IP packets from Host 1 and Host 2.

| No. | Time | Source | Destination | Protocol | Info |
|-----|-------------|-----------------|-------------------|----------|---|
| 335 | 1272.054499 | 192.168.0.2 | 224.0.0.5 | OSPF | Hello Packet[Malformed Packet] |
| 336 | 1276.054676 | 192.168.0.2 | 224.0.0.5 | OSPF | Hello Packet[Malformed Packet] |
| 337 | 1278.578581 | 192.168.1.5 | 192.168.0.1 | UDP | Source port: phoenix-rpc Destination port: smc-http |
| 338 | 1278.578642 | 192.168.0.1 | 192.168.1.5 | ICMP | Destination unreachable (Port unreachable) |
| 339 | 1280.054851 | 192.168.0.2 | 224.0.0.5 | OSPF | Hello Packet[Malformed Packet] |
| 340 | 1283.577191 | D-Link 81:23:2b | 00:00:00 00:00:01 | ARP | Who has 192.168.0.2? Tell 192.168.0.1 |

Frame 337 (66 bytes on wire, 66 bytes captured)
Ethernet II, Src: 00:00:00 00:00:01 (00:00:00:00:00:01), Dst: D-Link 81:23:2b (00:26:5a:81:23:2b)
Internet Protocol, Src: 192.168.1.5 (192.168.1.5), Dst: 192.168.0.1 (192.168.0.1)
User Datagram Protocol, Src Port: phoenix-rpc (3347), Dst Port: smc-http (6788)
Data (24 bytes)
Data: 1111111111111111A5A5A5A5A5A5A5AAAAAAAAAAAAAAAAAAAA
[Length: 24]

```
0000 00 26 5a 81 23 2b 00 00 00 00 01 08 00 45 00  6Z.#+.. ....E.
0010 00 34 00 00 40 00 3f 11 b9 62 c0 a8 01 05 c0 a8  .4..@.? .b.....
0020 00 01 0d 13 1a 84 00 20 cd 37 11 11 11 11 11 11  ....7.....
0030 a5 a5 a5 a5 5a 5a 5a aa aa aa aa aa aa aa aa aa  ...ZZZ.....
0040 aa aa
```

NC header
Encoded data
Original data

```
root@fpganet2:~# java client
Please enter an 8-byte word (16 hexadecimal digits)
send pkt
```