Scalable Emulation of IP networks through Virtualization

Amit Kucheria

Masters Thesis Defense
University of Kansas, Lawrence
May 9th 2003

Committee:
Dr. Douglas Niehaus (Chair)
Dr. Victor Frost
Dr. Jeremiah James
Talk Content

• Introduction
• Virtual Network Framework
• Design of Virtual Network Elements
• Evaluation of Virtual networks
• Summary
Introduction – Problem

• Ubiquitous IP networks – Data, Video & Voice
• Need to study and test new protocols
• Large scale networks
• Current testing methods:
  • Simulation
  • Physical testing
Introduction – Existing methods

• Simulation
  • ns2, OPNET
  • Side-effects of OS interactions ignored
  • Management complexity ignored
  • Change in focus

• Physical Testing
  • Equipment/Infrastructure costs
  • Erroneous extrapolations of small tests
Introduction - Goals

• Design & Implement Network Emulation Framework to solve current problems
• Test with realistic network loads
  • Generated by real utilities (e.g. ttcp, ftp, telnet, etc.)
  • Synthesized loads through Netspec
• Compare results with results from physical network
Introduction – Virtual Network Elements

• Definition:
  A virtual network element (VNE) is a software object that emulates the functions of network elements such as hosts and routers.

• Modules inserted into the Linux network protocol stack transparently

• New layer added to protocol processing sequence: Virtual Network layer (VNL)
Introduction - VNE

- Simple application of virtual network
- Virtual network traffic multiplexed over physical interface(s)
- VNL handles mux/demux
Virtual Network Framework (VNF)

- Three basic elements of a network: host, router & link
- Host and Router emulated by virtual host and virtual router code in VNL
- Virtual link implemented using link throttling techniques of Linux traffic control
- $\Sigma$ throughput(virtual elements) $\leq \Sigma$ throughput(physical interfaces)
VNF: Design Considerations

- Socket-layer compatibility
- Creation/Deletion/Configuration
- Arbitrary mapping of virtual elements to physical hosts
- Virtual routing decisions
- Network emulation ability
VNF: Architecture

- Multiple virtual hosts and virtual routers share VNL
- ETH_P_KUVNET is the packet type
- Each of the virtual devices can have a queue attached
VNF: Capabilities

- Multi-homed (virtual) hosts
- Split subnets across physical machines
- Supports almost arbitrary mapping of virtual elements to physical hosts through subnet maps
- Subnet map identical to routing table
- VNL inserts a new header: VNET header between IP and MAC header
VNF: Capabilities: Split-subnet

- Flexibility in placing VNEs on physical machines
- Load/Application/Characteristic-based mapping
VNF: Example

- Depicts virtualization based on network application
- Depicts working of *split-subnet* mapping
- A, D: Servers
- A ↔ (E, F), D ↔ (B, C)
VNF: Example (continued)

Virtual routing table of virtual host A

<table>
<thead>
<tr>
<th>Destination</th>
<th>Gateway</th>
<th>Netmask</th>
<th>Flag</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1.0.0</td>
<td>129.237.125.1</td>
<td>255.255.255.0</td>
<td>R</td>
<td>Veth1</td>
</tr>
<tr>
<td>0.0.0.0</td>
<td>10.1.0.254</td>
<td>0.0.0.0</td>
<td>G</td>
<td>Veth1</td>
</tr>
</tbody>
</table>

Subnet map of virtual hosts

- **Testbed 1**: A, B, C
  - A: 10.1.1.1 veth1
  - B: 10.1.1.2 veth2
  - C: 10.1.1.3 veth3
- **Testbed 2**: 10.1.0.254 vpost1
  - 10.2.0.254 vpost2
- **Testbed 3**: D, E, F
  - D: 10.2.1.1 veth1
  - E: 10.2.1.2 veth2
VNF: Example (continued)

- Alternative mapping of VNE to physical machines
Design of Virtual Network Elements

• Implemented as Linux network device driver
• Configured through ioctl()s
• Netspec-based configuration
• Shows virtual interface statistics through tools such as ifconfig, ip, etc.
• Supports packet capture tools such as tcpdump
VNE design: Virtual Host

- Has an IP address
- Virtual routing table contains gateway entry
- Subnet map table contains location of virtual router emulating the gateway
- Acts as source or sink
- Socket applications bind to it
VNE: Virtual Router

- Each port has an IP address
- Virtual routing table contains entries to other routers or to subnets
- Subnet map table contains location of virtual router and subnets
- Only a gateway for the packets
- Socket applications typically don’t bind to it (exception: RSVP daemon)
VNF: Example (continued)

<table>
<thead>
<tr>
<th>Destination</th>
<th>Gateway</th>
<th>Netmask</th>
<th>Flag</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1.2.0/24</td>
<td>129.237.125.2</td>
<td>255.255.255.0</td>
<td>i</td>
<td>Vport1</td>
</tr>
<tr>
<td>10.2.0.0/24</td>
<td>129.237.125.2</td>
<td>255.255.255.0</td>
<td>i</td>
<td>Vport1</td>
</tr>
<tr>
<td>10.2.0.0</td>
<td>0.0.0.0</td>
<td>255.255.0.0</td>
<td>N</td>
<td>Vport2</td>
</tr>
</tbody>
</table>
Evaluation of Virtual Networks

- Control plane of emulated network remains the same as physical network
  - Identical software
  - Identical signaling costs
- Need to confirm verity of data plane results
- Results of Physical tests vs. Emulation tests
- Diffserv and Intserv networks used for exercising VNF
Evaluation of Virtual Networks

- Diffserv relies on Linux traffic control ($tc$), hence works with VNE with minor modifications
- Intserv relies on $tc$ and RSVP signaling, RSVP required some porting to understand virtual routing
DiffServ - Network Topology (9 elements)

- Link bandwidth – 100Mbps in access network & 10Mbps inside core
- Routers – Emulated on high speed Pentium III, 1GHz, 1GB RAM Linux systems
Diffserv - Network Topology (9 elements)

• Used to validate working of Diffserv
• Throughput of ‘Test’ stream measured in presence of background RT and BE load
• Netspec-generated CBR traffic using UTIME patches
• tcpdump output captured at source and sink, merged and diff’ed
## Diffserv - Network Test parameters

| Traffic          | BG-BE traffic = 4-10Mbps  
|                  | BG-RT traffic = 0-10Mbps  
|                  | Test CBR traffic = 4Mbps  
| Diffserv Parameters (core routers) | Real time AF class = 6Mbps  
|                  | Best Effort class = 4Mbps  
|                  | HTB queuing discipline    
| Data Plane       | Throughput                
|                  | Delay                     |
Diffserv – Throughput Comparison (9 elements)

Physical network results
Diffserv – Throughput Comparison (9 elements)

Virtual network results
Diffserv – Throughput Comparison (9 elements)

Physical network zoomed results
Diffserv – Throughput Comparison (9 elements)

Virtual network zoomed results
Diffserv – Delay Comparison (9 elements)

**Physical**
- Uncongested: 2-3 ms
- Congested: 14-18 ms

**Virtual**
- Uncongested: 2-3 ms
- Congested: 15-17 ms
Physical Network Topology – Diffserv - 17 elements
Virtual Network topology – Diffserv – 17 elements

- Physical testing not performed due to shortage of machines
- Ideal case for using VNF
- Need to compare Diffserv properties observed in Physical networks with those observed in Emulated networks
Virtual Network topology – Diffserv – 17 elements
Virtual Network topology – Diffserv – 17 elements

• One greedy customer does not affect other customers of network
• Throughput of 2 ‘Test’ streams measured in presence of background RT and BE load from their respective networks
• `tcpdump` output captured at source and sink, merged and diff’ed
# Diffserv - Network Test parameters

| Traffic          | • BG-BE traffic = 2-6Mbps  
|                 | • BG-RT traffic = 0-6Mbps  
|                 | • Test CBR traffic = 4Mbps  
| Diffserv Parameters (core routers) | • 2 Real time AF classes = 6Mbps each  
|                 | • Best Effort class = 4Mbps  
|                 | • **HTB** queuing discipline  
|                 | • 6 Mb RT traffic is threshold (Test + BG)  
| Data Plane       | • Throughput  
|                 | • Delay  

### Diffserv – throughput Results (17 elements)

- #1-6 show equal traffic on both networks
- #7-8 show E2E #2 being greedy
- One greedy customer does not affect others in Diffserv

<table>
<thead>
<tr>
<th>#</th>
<th>E2E</th>
<th>BG-RT (Mbps)</th>
<th>BG-BE (Mbps)</th>
<th>throughput (Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>3.9901</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>3.9901</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>3.9904</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>3.9901</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1.5</td>
<td>2</td>
<td>3.9901</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1.5</td>
<td>2</td>
<td>3.9901</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1.5</td>
<td>4</td>
<td>3.9900</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1.5</td>
<td>4</td>
<td>3.9897</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>3.4735</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>3.4648</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>3.4752</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>3.4551</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1.5</td>
<td>2</td>
<td>3.9901</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>3.4799</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1.5</td>
<td>4</td>
<td>3.9901</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>3.4591</td>
</tr>
</tbody>
</table>
Diffserv – Throughput Comparison (9 elements)

Physical network zoomed results
Diffserv – Throughput Results (17 elements)
Diffserv Data Plane Evaluation – Results

- Diffserv behaves similarly in physical and emulated (virtual) networks
- Results of throughput/delay tests on emulated network similar to those of physical network
- Very minor changes to code to get Diffserv to work with VNEs
Emulating Intserv networks

• Emulated network is identical to Diffserv network
• Diffserv traffic classes replaced by RSVP daemon which does dynamic resource reservation
• RSVP daemon modified:
  • To understand virtual routing
  • To enable many instances to run on a physical machine bound to specific VNEs
Emulating Intserv networks

- Intserv network successfully emulated
- Results on Physical network not reproducible for multiple iterations of tests
  - RSVP daemon uses CBQ
  - Linux CBQ implementation tries *estimation* to schedule packets, does not give consistent results
  - HTB implementation for RSVP non-trivial
- Data plane could not be verified
- Demonstrates clean interface of VNF that allows complex applications to use it
Limitations of VNF

- Sum of throughputs of VNEs on a physical machine must be less than sum of throughput of all physical interfaces; overcome using virtual time techniques introduced by ProTEuS

- Doesn’t allow ‘connected’ NEs to be emulated on same physical machine if packet needs to pass through queuing code; can be overcome by modifying queuing code
Summary

• VNF designed and implemented
• Tested with non-trivial IP networks such as Diffserv and Intserv networks
• Programming model allows easy ‘porting’ of applications to work with VNF
• Larger Diffserv networks successfully emulated
• Intserv networks emulated functionally, but data plane could not be verified
Thank You