1-10 Gbps IPv6 Programmable IDS/IPS

Livio Ricciulli
livio@force10networks.com
(408) 835-5005

*Supported by the Division of Design Manufacturing and Industrial Innovation of the National Science Foundation (Awards #0339343, 0521902) and the Air Force Rome Laboratories.
Open architecture to leverage open source software
   - More robust, more flexible, promotes composability
   - Directly support Snort signatures
   - Abstract hardware as a network interface from OS prospective

Retain high-degree of programmability
   - New threat models (around the corner)
   - Extend to application beyond IDS/IPS

Line-speed/low latency to allow integration in production networks
   - Unanchored payload string search
   - Support analysis across packets
   - Gracefully handle state exhaustion

Hardware support for adaptive information management
   - Detailed reporting when reporting bandwidth is available
   - Dynamically switch to more compact representations when necessary
   - Support the insertion of application-specific analysis code in the fast path
Monitoring 10GE

Ariën Vijn
arien.vijn@ams-ix.net
Introduction

- AMS-IX
  - Internet Exchange point
  - Facilitate BGP peering
Introduction

- AMS-IX
  - Internet Exchange point
  - facilitates BGP peering
- Members connect border routers to our switch platform
Introduction

- AMS-IX
  - Internet Exchange point
  - facilitates BGP peering
- Members connect BGP routers to our switch platform
- Everyone can peer with everyone else
Introduction

- AMS-IX
  - Internet Exchange point
  - facilitates BGP peering
- Members connect BGP routers to our switch platform
- Everyone can peer with everyone else
- Layer 1&2 platform
  - Glimmerglass fiber switches
  - Foundry Networks ethernet switches
Introduction - numbers

- Peak traffic: 136Gbps
Introduction - numbers

- Peak traffic: 136Gbps
- Volume January 2006: 29100 Tbyte
- 240 members
- 384 member ports
- 64 IPv6 addresses
- 9 Multicast
- 38 operational 10GE member ports
Introduction - network problems

- Many issues can be analyzed by looking at multicast and flood traffic.
- But sometimes sniffing is required
  - Unicast
Introduction - network problems

- Many issues can be found by looking at multicast and flood traffic.
- But sometimes sniffing is required
  - Unicast
- Who is responsible?
  - Is it this AS?
Introduction - network problems

- Many issues can be found by looking at multicast and flood traffic.
- But sometimes sniffing is required
  - Unicast
- Who is responsible?
  - Is it this AS?
  - Is it that AS?
Introduction - network problems

- Many issues can be found by looking at multicast and flood traffic.
- But sometimes sniffing is required
  – Unicast
- Who is responsible?
  – Is it this AS?
  – Is it that AS?
  – Or… are we the ones to blame?
Example #1

- One MAC behind one port
  - Ensure stability
  - Port security feature
Example #1

- One MAC behind one port
  - Ensure stability
  - Port security feature
- Any other source MAC will be blocked
- We may shut the port
Example #1

- Any other source MAC will be blocked
- We may shut the port
- Syslog reporting is limited
  - Only shows the port and violating source MAC address

Port: 1/1
violating address: 00:00:5e:00:01:11
Example #1

- Any other source MAC will be blocked
- We may shut the port
- Syslog reporting is limited
  - Only shows the port and violating source MAC address
- Is this real?
  - Is this a member responsibility?
  - Is this the a bug in the switch?
Example #1

- Syslog reporting is limited
  - Only shows port and violating source MAC address
- Is it real or is it a bug?
  - Is this a member responsibility?
  - Is it the a bug in the switch?
- Provide proof to member
- Provide router vendors with clues to debug
Example #2

- Problem:
  - Lots of ethernet multicast traffic from router 1
  - IP payload is unicast
Example #2

- **Problem:**
  - Lots of ethernet multicast traffic from router 1
  - IP payload is unicast
- **Just before the problem started?**
  - ARP request from router 2 for router 3
Example #2

- **Problem:**
  - Lot of multicast traffic from router 1
  - IP payload is unicast

- **Just before the problem started?**
  - ARP request from router 2 for router 3
  - Multicast bit of sender MAC is set to one
Example #2

- Just before the problem started
  - ARP request from router 2 for router 3
  - Multicast bit of sender MAC is set to one
  - 10GE router 1 updates its ARP cache upon the ARP request of router 2
Example #2

- What happened just before that?
  - ARP request from router 2 for router 3
  - Multicast bit of sender MAC is set to one
  - 10GE router 1 updates its ARP cache upon the ARP request of router 2

- Sniffing revealed that this was a bit flip due to faulty switch hardware
- More sniffing was needed to determine which faulty switches and cards.
Monitoring 10GE

- We do not use the mirror port feature
  - Halves the ingress bandwidth on mirrored port
  - Port density
    - All 10GE ports are in use
- So what do we do?
Monitoring 10GE

- We do not use 10GE mirror port
  - Halves the ingress bandwidth on mirrored port
  - Port density
    - All 10GE ports are in use
- So what do we do?
- Each 10GE routers is connected via Glimmerglass fiber switch
Monitoring 10GE

- So what do we do?
- Each 10GE routers is connected via Glimmerglass fiber switch
- We use(d) Anritsu test equipment
  - Tester in ‘through’ mode
  - Acting as repeater
Monitoring 10GE

- Each 10GE routers is connected via Glimmerglass fiber switch
- We use(d) Anritsu test equipment
  - Tester in ‘through’ mode
  - Acting as repeater
- Testers are scarce, expensive lab resources
- We need dedicated 10GE monitor equipment on site
- Preferably a general purpose computer
10GE - capture cards

- Force 10 (MetaNetworks) MTP-10G card
  - Build for security applications
  - Snort in hardware
  - Not directly applicable
  - Acts like a line rate 10GE repeater
10GE - capture cards

- Force 10 (MetaNetworks) MTP-10G card
  - Build for security applications
  - Snort in hardware
  - Not directly applicable
  - Acts like a line rate 10GE repeater

- Open Design
  - Comes with sources
  - API (Verilog)
10GE - capture cards

- Open Design
  - Comes with sources
  - API (Verilog)
- Need to adapt the product to our needs
  - Layer 2
  - Technically possible
- Work in progress
  - Finished: end of May
  - We’ll use our Anritsu testers in the meanwhile
Flynn’s Computer Taxonomy

**MIMD**
- **Instructions**
- **Memory**
- **Processor**
- **Data**

**MISD**
- **Alert**
- **Reduction Network**
- **P0**
- **P1**
- **...**
- **Pn**
- **Data**
- **Instructions**

**SISD**
- **Instructions**
- **Memory**
- **Processor**
- **Data**

**SIMD**
- **Alert**
- **Reduction Network**
- **P0**
- **P1**
- **...**
- **Pn**
- **Data**
- **Instructions**
Layer-1 Filtering
Product Architecture

Latency = 1.3 \mu s

100Mb-10Gb

2-8M Concurrent Flows

100Mb-10Gb

PHY

RAM

IPS/IDS

Dynamic
rules

Runtime update

Static
rules

Synthesis + firmware update

Latency = 1.3 \mu s
Flexible Deployment Options

**Router/Switch**
- CPU
- IDS/IPS
- Inline
- IPS application
- Chain multiple cards inline for additional rule capacity

**Multiple Mirrors**
- Passive
- CPU
- IDS/IPS
- IPS application
- Chain multiple cards inline for additional rule capacity

**Mirror Port**
- Passive Inline
- CPU
- IDS/IPS
- Extend passive capacity
- Can hang multiple passive devices off 1 TAP or Mirror

**To other passive device**
- Multiple Mirrors
- Passive
- IPS application
- Chain multiple cards inline for additional rule capacity

- Mix of 1G and 10G
Stateful Content Inspection
Performance Comparison

Percentage of Alert Loss

Mbps

% of alert loss

-20.00%

0.00%

20.00%

40.00%

60.00%

80.00%

100.00%

0 500 1000 1500 2000 2500 3000

darpa no MTP
web1 no MTP
web2 no MTP
darpa with MTP
web1 with MTP
web2 with MTP
**Intuitive Management Tools**

- **Interface**
  - Card operates as a standard NIC
  - Reuse all existing Unix-based utilities/applications
  - Policies implemented rule by rule for block, forward, ignore and capture
IPv6 Security Hardware

- IPv6 options provide a covert channel
  - Ex. Joe 6 pack ([http://people.suug.ch/~tgr/misc/j6p-1.0.tar.gz](http://people.suug.ch/~tgr/misc/j6p-1.0.tar.gz)) uses IPv6 Destination option for transport

- Want to see what are IPv6 options used for (for example source routing)
  - Extend hardware payload match semantics to IPv6 header

- Tunneling
  - Want to inspect headers of multiple tunnels
Additions to IPv6 API

- 8-bit “parse” value indicating which section of the packet is being clocked in
  - Unknown
  - IPV4 = 0x4
  - Payload = 0xFE
  - TCP = 0x6
  - ICMPV4 = 0x1
  - UDP = 0x11
  - IPV6 = 41
  - Routing = 43
  - Fragment = 44
  - Destination = 60
  - Authentication = 51
  - Security Payload = 50
  - ICMPv6 = 58
  - Hop by Hop = 0

- Counters
  - Tunnel “tcnt” counter
  - Length offset within section pointed to by “parse”
User-level programmability

- Define API to let user write ad-hoc wire-speed code
- Add user modules to synthesis flow and share reduction network
- Architecture provides determinism
  - It either fits or it does not fit in the FPGA
  - It either meets timing or does not meet timing
  - Load/store network processing much harder to predict
- **Rst**: The signal is a pulse indicating the start of a packet.
- **Enable**: This signal asserts that the data is valid.
- **Data**: This 128-bit signal conveys the packet data.
- **Offset**: This 11-bit signal conveys which segment of the packet is being clocked in.
- **Inpacketoff**: This 8-bit signal conveys the byte offset of the first payload byte.
- **Dsize**: This 16-bit signal conveys the size (in bytes) of the payload.

- **Cin**: This 16-bit signal conveys state information of the flow of the packet being received.
- **Matchresult**: Capture the packet and provide it to the host through the PCI bus.
- **R**: This signal indicates to the system to store the packet in temporary match memory.
- **F**: This signal has the same effect as **matchresult** but, in addition, allows linking the packet currently being matched with other packets previously matched with the **r** signal.
- **B**: This signal causes the forwarding of the packet to be interrupted.
- **C**: This 16-bit signal is the state vector to be stored for the flow represented by the current packet.

- **Rulenum**: This 32-bit value is stored as trailing data to the packet.
- **Init**: This input signal is asserted when the MTP10G is reset.
- **Program**: Asserted whenever there is valid data to be written in the user-defined configuration registers.
- **Address**: This 18-bit signal indicates the address of the registers or memory being referenced by the host.
- **data_conf_in**: This 32-bit signal is the data to be written to the registers or memory.
- **data_conf_out**: This 32-bit signal is the data read from the registers or memory.
- **Magic**: This read-only 32-bit value is provided to the host for management purposes.
Stateful Support

Temporary Memory

Match Memory

Packet 0 0
Packet P1 P2
Packet P1 P2
Packet 0 0
Packet P1 P2
Packet P1 P2
Packet P1 P2
Packet P1 P2
module cam8(init, rst, clk, cnfclk, data, offset, program, address, data_conf_in, data_conf_out, magic, matchresult, enable, r, c, f, b, Cin, dsize, inpacketoff, rulenum);

always@(posedge clk or posedge rst) begin
if (rst) begin
    // Initialize local registers
    b <= 0;
    r <= 0;
    f <= 0;
    c <= 0;
    matchresult <= 0;
    magic <= <user ID>;
end
else begin
    // User function
    b <= <user function>
    r <= <user function>
    f <= <user function>
    c <= <user function>
    matchresult <= <user function>
end
end

always@(posedge cnfclk or posedge init) begin
if (init) begin
    // Initialize local control registers
end else if (program) begin
    case (address)
    18'h2xxx: <local control register> <= data_conf_in;
    18'h3xxx: <local control register> <= data_conf_in;
    endcase
else begin
    case (address)
    18'h2xxx: data_conf_out <= <local control register>;
    18'h3xxx: data_conf_out <= <local control register>;
    endcase
end
endmodule
memory mem(.c1(clk),.a1(dstp[15:0]),.di1(newval),.do1(oldvalout),.w(write),.c2(cnfc1k),.a2(address[15:0]),.do2(valout));

always@(posedge clk)
begin
    if(offset==1) begin
        proto<=data[7:0];
    end else
    if(offset==2 && (proto==06 || proto==17)) begin
        dstp<=data[31:16];
    end else
    if(offset==4 && dstp!=0) begin
        newval<=oldvalout+1;
        write<=1;
    end else
    begin
        write<=0;
    end
end
Reuse existing Open Source

```
root@r10:/MTP-Drivers
Cpu(s): 2.6% user, 0.0% system, 0.0% nice, 97.4% idle
Device Number: 0 -- Device Type: MTP-70 -- Firmware ID: b

Hardware Interfaces

<table>
<thead>
<tr>
<th>Total Packets</th>
<th>CHO Bottom UP</th>
<th>Rate/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP Packets</td>
<td>15837969</td>
<td>107K</td>
</tr>
<tr>
<td></td>
<td>15299920</td>
<td>101K</td>
</tr>
<tr>
<td></td>
<td>256407</td>
<td>1772</td>
</tr>
<tr>
<td></td>
<td>281562</td>
<td>4287</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Port #</td>
<td>pps</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>23</td>
<td>44779</td>
</tr>
<tr>
<td>80</td>
<td>80</td>
<td>8916</td>
</tr>
<tr>
<td>22</td>
<td>22</td>
<td>4864</td>
</tr>
<tr>
<td></td>
<td>29776</td>
<td>2403</td>
</tr>
<tr>
<td></td>
<td>14353</td>
<td>2106</td>
</tr>
<tr>
<td></td>
<td>20071</td>
<td>2057</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>2036</td>
</tr>
<tr>
<td></td>
<td>14354</td>
<td>1914</td>
</tr>
<tr>
<td></td>
<td>19255</td>
<td>1272</td>
</tr>
<tr>
<td></td>
<td>30797</td>
<td>1243</td>
</tr>
<tr>
<td></td>
<td>20541</td>
<td>1183</td>
</tr>
<tr>
<td></td>
<td>3049</td>
<td>1155</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CH1 Top</th>
<th>UP</th>
<th>Rate/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>3711567</td>
<td></td>
<td>116K</td>
</tr>
<tr>
<td>3651557</td>
<td></td>
<td>115K</td>
</tr>
<tr>
<td>36547</td>
<td></td>
<td>149</td>
</tr>
<tr>
<td>22365</td>
<td></td>
<td>35</td>
</tr>
<tr>
<td>1099</td>
<td></td>
<td>97</td>
</tr>
</tbody>
</table>

```

h=help  z=stop  p=port
- P10 PCI Card (10 GbE interface)
  - High speed PCI card in 1U chassis
  - Wire-speed stateful deep packet inspection; 20G-in/20G-out
  - 650 static rule capacity; 65 dynamic rules; (currently being increased);
  - 8 million concurrent flows

- P1 PCI Card (GbE interface)
  - High speed PCI card in 1U chassis
  - Wire-speed stateful deep packet inspection; 2G-in/2G-out
  - 1000 static rule capacity; up to 200 dynamic; (currently being increased);
  - 2 million concurrent flows

- P1/P10 Appliance
  - 1U host embeds a P1 or P10 PCI card
  - Software and drivers pre-installed and pre-configured
Summary

- Extremely low latency design enables a wide variety of deployment options
- Leverage Open Source software
- 1G and 10G available today
- Processing paradigm lends itself to ad-hoc application level programmability

Livio Ricciulli
livio@force10networks.com
(408) 835-5005
www.metanetworks.org
Thank You