Network Monitoring, Visualization

Gigabit Kits Workshop
(January 10, 2001)

Ken Wong, Eileen Kraemer*, Jon Turner
Washington University and University of Georgia*

NSF ANI-9714698
http://www.arl.wustl.edu/arli/projects/nmvc

Topics

- NMVC Project Overview
- Monitoring
  - Switch Monitor (swmon)
  - IP Monitor (kipmon)
  - Denial of Service Example
- Visualization
- Plans
Main Participants

Arl Staff
  Jyoti Parwatikar

Washington University Graduate Students
  Anshul Kantawala
  Samphel Norden
  Ed Spitznagel

University of Georgia Graduate Students
  Piyush Burte
  Sneha Kadandale

Goals

- Network management testbed
  - Highly configurable service fabric
  - High-speed, programmable monitoring and filtering
  - Flexible data analysis

- Multilayer system management tools
  - Enable management across multiple layers of a distributed system
  - Provide semantic richness
  - Enable direct control and algorithmic feedback to system components
Mechanisms for NMVC Implementation

- Flexible, high-performance probes
  - High-performance network probe using APIC-based platform
  - Directed polling and multicast for remote probing
  - General software probe and agent architecture

- Plugable kernel modules (Crossbow)

- Event filtering and data fusion/reduction

- Extensible visualization and control system
  - Multilayer graphic representations of system components
  - Direct access to published attributes (read/write)

Year 2000 in Review

- Switch Monitor swmon (Linux)
  - Version 0 in Spring 2000
  - Fixed some bugs
  - Works with simple Java GUI derived from UGA GUI

- IP Monitor kipmon
  - Version 0 almost done

- Experimental Java GUI
  - Converted from AWT to Swing
  - Architecture matches switch monitor
  - Designing
swmon Interface

- `swmon_begin/end` (a NOCd session)
- `swmon_open/close` (a switch context)
- `swmon_on/off` (a meter)
  - Throughput (Input Port, Output Port, VC)
  - Cell Discard (Input Port, Output Port)
- `swmon_read/write` (a VC entry)
- `swmon_set VC priority`

* Special purpose

Switch Monitor (swmon)
IP Monitoring With Plugins

- Experience with basic IP monitoring
- Experience with router plugins
- Step 0: Meter concept matches CB framework
  - Use Meter == Load plugin if not loaded; create instance; add filter
  - Start Meter == Register instance
- Eventually:
  - Use CB filters (or FPX?) for gross filtering
  - Plugin+Daemon contains more general filter
  - Additional system features (Ex: IP monitoring model, language, ???)

IP Monitoring Example

- Monitor at router 192.168.20.1
  - Use a packet counter meter for UDP packets destined for 192.168.5.2/128, port 5050
  - Use another packet counter meter for UDP packets destined for all other ports at 192.168.5.2/128
- Also, monitor at router 192.168.5.1, ...
**IP Monitor (kipmon)**

At 192.168.20.1

Use cntr.o int=* pro=udp src=* dst=(192.168.5.2/128,5050)

Use cntr.o int=* pro=udp src=* dst=(*,5050)

Start all

Stop 2

Probe 192.168.5.1 . . .

- User-space daemon like NOCd that understands meter abstraction (CM wrapper + TCP msgs)

---

**Application Examples**

- Rapid network deployment
- Visual network exploration
- Dynamic policy-based resource allocation
- End-to-end connection tracing
- Network / System security
Denial of Service Attack

- Attackers attempt to prevent legitimate users of a service from using that service
- Examples
  - Flood network: Push out legitimate traffic/crash host
  - Disrupt connections to/from a machine
  - Disrupt/prevent service to a system or user
- Typical Distributed Attack Mode
  - Enumerate potential recruits
  - Identify vulnerable host services (Ex: root sendmail)
  - Recruit host
  - Repeat above until critical mass and attack victims

Distributed DoS

<table>
<thead>
<tr>
<th>Attack</th>
<th>Client</th>
<th>Nodes</th>
<th>1 Node: 1000s ppm</th>
<th>1000s of Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Floods/Attacks (Trino, TFN2K, ...)</td>
<td>Victims</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stealth: ICMP piggyback Encrypt Cmnds</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Wong, Kraemer and Turner 1/11/01
Flood Response

- Record baseline usage
  - Monitor and save network traffic characteristics (Ex: volume by protocol by period of day) in history DB
- Install alarms (multiple levels?) at key points (edge routers, key switch ports)
  - Alarm levels based on baseline
- Alarm may activate:
  - Install blocking filter near victim
  - Begin attack path discovery (Install other meters)
- Integrate with host-based recognition and response

GUI Features

- Java
- Topology at multiple network layers
- Virtual Networks (subgraph aggregation)
- Find meters and meter slots
- Control knobs
- Create, compose and control meters
- Select visualizations
- Store meter values for later visualization
Application Problem Tracing

Video received at GIT is bad even though outgoing video stream from Princeton is good. Is there something wrong at WU?

Aggregated Nodes

<table>
<thead>
<tr>
<th>File</th>
<th>Vis</th>
<th>View</th>
</tr>
</thead>
</table>

Network Link

Wong, Kraemer and Turner 1/11/01
Show Meter Slots

IP Meters and Meter Attributes
Current Plans

- Experience: IP Flow Monitoring Plugins/SPC
- Virtual Switch Interface
  - Discovery, Initialization (Boot)
  - Control
- Java GUI
  - Initialization (Boot)
  - Topology diagrams (Link, Network)
  - Configuration management
  - Integration with metering system
- Architecture for MultiService Router
  - Programming paradigm
Next Generation Management Tools

- **Integration**: Of management across all system layers
- **Correlation**: Of events in physical/abstract space
- **High-Speed**: Monitoring, flexible filtering, data analysis
- **Programmability**: Of management services
- **Automation**: Of management functions (configuration, troubleshooting, ...)
- **Navigation**: Between entities at different abstraction levels
- **Direct Control**: With algorithmic feedback to system components

Challenges

- **Programming paradigm**
  - Minimize cognitive load
  - Rich semantics
  - Composable modules
- **Management code integrity**
  - Fault prevention
  - Fault isolation
- **Gateways to legacy systems**
- **Management of management objects**