Extensible Network Security Services on Software Programmable Router OS

David Yau, Prem Gopalan, Seung Chul Han, Feng Liang

System Software and Architecture Lab
Department of Computer Sciences
Purdue University

Motivations

- Value-added services router technologies
- Need for secure network infrastructure
  - attacks are easy (DDoS, address spoofing, network intrusions, ...)
  - need to detect/trace/counter attacks in flexible ways
    - protect innocent users, prosecute criminals, etc
Existing Networks

Value-added Services Networks
Project Goals

- OS prototype (CROSS) for value-added services routers
- Security services to be dynamically injected into routing infrastructure

Router OS Challenges

- Heterogeneous users
  - needs, priorities, purchased shares
- Untrusted programs
  - greedy, buggy, malicious, ...
- Diverse resources
  - space-shared, time-shared
- Diverse resource bindings
  - multi-processes, multi-threads, multiplexed threads
The CROSS Approach

- Virtualized router resources
  - virtual machines
- Orthogonal fine-grained allocations
  - Resource Allocation objects
- Flexible/scalable packet classification
  - resource binding, per-flow processing
- Efficiency, modularity, configurability

Resource Virtualization

- Hierarchical scheduling
  - virtual machines with different APIs
  - user allocations on demand
- Target resource types
  - CPU time
  - network bandwidth
  - memory pool capacity (virtual memory)
  - disk bandwidth
Resource Abstraction

- Kernel Resource Allocation objects
- Independent/orthogonal objects
  - relative to resource consumers
- Flexible bindings to resource consumers
  - shared binding
  - dynamic binding (with run-time information)
  - configurable parameters

Resource Allocations
Packet Forwarding

- Three possibilities
  - Active program dispatch
    - Trusted (kernel thread), untrusted (user process)
  - Per-flow processing
    - Subscribed by dispatched router programs
    - Security processing, application-level routing
  - Cut-through fast path
    - Minimal delay

Packet forwarding decision

- Based on packet header information
- Packet classification
  - Scalable to many dimensions
  - Scalable to many classification rules
  - Flexible
    - Support multiple and least-cost matches
Cross Forwarding Paths

Example Scheduler: CPU

- Hierarchical partitioning using fair service curves [MMCN 2001]
- Decoupled delay and rate allocation
  - Good for low delay and low rate applications
- Solution to priority inversion
  - Lock contention and client/server interaction
- Performance
  - Rate/delay guarantees, proportional sharing, minimized unfairness
Service Curve

CPU time promised vs. Time since thread wakes up

- Convex curve
- Linear curve = rate (0.3)
- Concave curve

CPU Sharing Hierarchy

- CPU
  - VM foo
  - VM bar
  - VM doe
  - User allocation A
  - User allocation B

bind process
thread
System Implementation

- Extension to Solaris 2.5.1/Linux 2.2.x/2.4
- Deployed on UltraSPARC/Pentium network
  - Ethernet, Fast Ethernet, Myrinet
- Modular subsystems with well-defined interfaces
- Simple command interfaces to launch legacy applications

Basic Costs

- Resource Allocation control
  - create
  - delete
  - bind/unbind
- Function dispatch
  - thread: about 145 microseconds, low variance
  - process: 0.77 to 1.1 ms, application-dependent
### Resource Allocation Costs (microseconds)

<table>
<thead>
<tr>
<th>Operation</th>
<th>kernel</th>
<th>user</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bind</td>
<td>4.8</td>
<td>9.0</td>
</tr>
<tr>
<td>Unbind</td>
<td>2.4</td>
<td>6.6</td>
</tr>
<tr>
<td>Create + delete</td>
<td>15.4</td>
<td>19.6</td>
</tr>
</tbody>
</table>

### Packet Classification

- Five dimension
  - exact, prefix, range, wildcard
- Database size up to 256 K rules
- Average lookup cost of 7.8 microseconds
  - 1.1 Gb/s for 1000 byte packets
- Add/delete 10.8/14.9 microseconds
  - 67,000 updates per second
Packet Classifier Performance

Example Service: SmartTrack

- Track back to sources of DDoS attacks
- When an attack is detected, launch a CROSS program detective at affected routers, and recursively trace back upstream
- Enable a victim domain to take various actions in response to an attack
**SmartTrack Mechanism**

- Attack monitors deployed at edge router
- Check for suspicious packet/statistical behaviors of incoming traffic
- When attack detected, dispatch a detective program to perform more careful traffic analysis
- Propagate detectives to upstream routers
- Result: identify and isolate attackers

**SmartTracking**

- I am attacker!
- Manager
- Stop!

Monitoring suspicious packet type/statistical behavior of traffic at edge router
Attackers send malicious packets
Dispatch detective program for detail analysis
Propagate detective to upstream routers
Identify and isolate attacker
Further information

- System Software and Architecture Lab
  - http://ssal.cs.purdue.edu
  - CROSS paper: Resource Management in Software-programmable router OS (IEEE JSAC, March 2001)

- People
  - David Yau, project director (yau@cs.purdue.edu)
  - Prem Gopalan (gopalapk@cs.purdue.edu)
  - Seung Chul Han (han@cs.purdue.edu)
  - Feng Liang (liangf@cs.purdue.edu)

CPU/network Scheduling

- Network respond application
  - driven by received packets
  - do some CPU computation, send some network data out

- Total delay budget of 3.5 seconds
  - CPU one second, network 2.5 seconds
  - CPU two seconds, network 1.5 seconds

- Allow both rate and delay compositions
## Rate Composition

<table>
<thead>
<tr>
<th></th>
<th>Udpburst CPU rate</th>
<th>Greedy CPU rate</th>
<th>Achieved bandwidth (Mb/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5%</td>
<td>95%</td>
<td>95%</td>
<td>3.6</td>
</tr>
<tr>
<td>10%</td>
<td>90%</td>
<td>90%</td>
<td>7.8</td>
</tr>
<tr>
<td>15%</td>
<td>85%</td>
<td>85%</td>
<td>9.8</td>
</tr>
<tr>
<td>20%</td>
<td>80%</td>
<td>80%</td>
<td>9.8</td>
</tr>
</tbody>
</table>

## Delay Composition (microseconds)

<table>
<thead>
<tr>
<th>Run</th>
<th>Mean CPU delay</th>
<th>S. d. CPU delay</th>
<th>Mean net delay</th>
<th>S. d. net delay</th>
<th>Mean total delay</th>
<th>S. d. total delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.06</td>
<td>0.007</td>
<td>2.31</td>
<td>0.136</td>
<td>3.39</td>
<td>0.144</td>
</tr>
<tr>
<td>2</td>
<td>2.00</td>
<td>0.096</td>
<td>1.48</td>
<td>0.136</td>
<td>3.49</td>
<td>0.172</td>
</tr>
</tbody>
</table>
Disk Scheduling

- Program: search through an input file sequentially for some pattern
- Two groups of 10 processes each
  - Group one: reading 65,588 kbytes, with allocation of rate 10
  - Group two: reading 55,789 kbytes, with allocation of rate 20
- Equal CPU allocations, disk placement not controlled

Proportional Disk Sharing

![Graph showing proportional disk sharing performance]
Memory Scheduling

- Footprint application
  - repeatedly touch a set of $n$ distinct pages
- Result summary [JSAC 2001]
  - isolation properties
  - utilization of over-reserved pages
  - reclaim of reserved pages

Related Work

- Router Plugins (Washington U)
  - extensibility, quick resource binding through gates
- Extensible Router (Princeton)
  - kernel built from scratch, path abstraction
- Bowman (Georgia Tech/U Kentucky)
  - Posix user-level implementation for portability
Related Work (cont’d)

- Active node with ANTS (MIT)
  - externally certified program capsules
- Flexible end-system scheduling
  - Resource Containers (Rice)
  - Software Performance Units (Stanford)
  - Reservation Domains (AT&T)

Conclusions

- Resource management important for software-programmable routers
- Presented system prototype as solution step
  - packet classification
  - router program dispatch
  - unified and orthogonal resource abstraction
  - schedulers for major resource types
Resource Allocation

Scope of System Integration

- CPU scheduling
  - threads and processes
- Network scheduling
  - packets in Stream buffers
- Memory scheduling
  - page frames, MMU reference bits
- Disk scheduling
  - buffer header structures
CPU Sharing Hierarchy

Disk Reservation Binding
Packet Processing Paths

Packet Classifier Memory

[Diagram showing packet processing paths and packet classifier memory usage graph]

Memory Usage vs. Number of Filters graph

- Memory Usage
- Number of Filters

20
CROSS Processing Paths

Resource Allocation API

- Create/delete
  - named by object system-wide key
- Bind/unbind
  - affect calling thread/process
  - key to fine-grained resource management
- Control
  - change scheduling parameters, owner, ...
- User-level access through pseudo-device
**System Integration**

- Leverage against Solaris gateway OS
  - support for existing application
  - immediate access to software development platform
- Implication
  - need to work with existing Solaris abstractions
  - threads/processes, stream buffers, page frames, buffer header structures, ...

**Memory Scheduler**

- Guaranteed share per allocation
  - minimum number of page frames that allocation can map simultaneously
- Guaranteed-share scanner algorithm
  - consider pages for replacement in decreasing over-allocation order
  - second chance to referenced pages
    - allow reserved but unused pages to be utilized
Memory Allocation

Disk Scheduler

- Provide proportional sharing
  - conflict with efficiency goal to minimize seek time overhead
  - notion of eligibility to balance between the two goals, using tolerance parameter
- Integrated with file systems
  - problem: applications do not access disk directly!
File System Disk Access

- Resource Allocation bound to resource principals
- Resource principals do read/write/mmap system calls
  - Disk accesses avoided unless file system page faults
- Page faults occur in interrupt context!
- Solution: Association Map on vnode/offset

Association Map

- Resource principal
- Upper-half file calls: read/write/mmap
- Vnode/offset to allocation mapping
- Association map
- Lookup allocation
- Disk server
- Vnode/offset
- Page fault
Bandwidth scaling with 64 and 128 byte packets