Experiences with Active Kernel Modules at Washington University

John DeHart
Washington University
jdd@arl.wustl.edu
http://www.arl.wustl.edu/arl/projects/ann.html

Questions: Who, What, Where

• Who has the information and access to be able to write the Active Code?
  –
• What level of performance is required to make the active code effective?
  –
• Where in the network are the resources to execute the code?
Specific Information Required

Data/User

Service/Application

Specific Operations

Network Specific Operations

All Three

Who Writes the Active Code

• End User - “classic” active networking
  – “Its my data, here’s how I want to manipulate it …”

• Application/Service Developer
  – “Its my application, I understand the format of it best, here’s some capabilities for manipulating it …”
  – e.g. Vendor X provides Active congestion control for its MPEG video conferencing system.

• Network Provider/Developer
  – “Its my router, I’ll give you these capabilities for manipulating your data …”
  – e.g. Alternate Routing and Topology based decisions.
Where is the Active Code Executed

- User Space Execution Environments
  - ANTs from MIT
  - CANES from Georgia Tech
  - PLAN from UPenn
  - ...
- Kernel Space Execution Environments
  - Scout from Arizona
  - ANN (Active Network Node) from WU
  - ...
- Hardware (e.g. FPGAs)
  - P4 from UPenn
  - PPX from WU
  - ...

“Taxonomy” of Active Functionality

- Generic Network
  - Routing
  - Congestion Control
  - Network Monitoring and Control (Traffic Data collection, Topology discovery, …)
  - Protocol Prototyping and Deployment
- Generic Data Manipulation
  - Encoding/Decoding/Transcoding
  - Encryption/Decryption
  - Congestion Control
- Specific Data Manipulation
  - Encoding/Decoding/Transcoding
  - Congestion Control
  - Application Specific Computation
Sample of Active Applications

- Everyone: Ping
- ActiveBridge - UPenn
- Universal Search Interface - Arizona
- MPEG video congestion control - Georgia Tech
- Wavelet video congestion control - WU
- Active Error Recovery - TASC/Umass
- Wireless
- Distributed Interactive Simulation
- ...

Requirements for WU AN Platform

- Code should be deployed automatically and on-demand to router
  - however this happens rarely!
- Simple security model we understand
- Fast enough for data-path applications
  - 155 Mbits/s in software
  - 1.2 Gbits/s with hardware support (FPGA)
Router Plugins

• Modular kernel software architecture developed at Washington U. and ETH Zurich

• Allows for
  – dynamically adding code modules called Plugins at run-time into the IP forwarding loop
  – binding plugins to individual IP flows

• Provides fast packet classification

Packet Processing in the WU Active Network Node
Principal Data Flows Through ANPE Kernel

- Standard processing for “plain” IP packets.
  - classification and routing, header processing, output queueing
- Active packets move through configured kernel plugins.
  - active function dispatcher passes packets to instances of plugin objects
  - instantiates objects or triggers download of plugin class, as needed
  - streamlined processing of SAPF packets using pre-established state
Active Congestion Control for Video

- Based on wavelet-based encoding method (WaveVideo)
  - up to hundreds of distinct frequency sub-bands (33)
  - low frequencies: general image definition
  - high frequencies: image details
- Active WaveVideo plugin
  - detects congestion
  - discards packets in high frequency sub-bands first
- Better performance than layered multicast
  - requires large number of multicast groups
  - reacts too slowly to congestion (several seconds)

WaveVideo Scaling

<table>
<thead>
<tr>
<th>Quality</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>33011 bytes (33)</td>
</tr>
<tr>
<td>30</td>
<td>25239 bytes (30)</td>
</tr>
<tr>
<td>27</td>
<td>17179 bytes (27)</td>
</tr>
<tr>
<td>24</td>
<td>9265 bytes (24)</td>
</tr>
<tr>
<td>21</td>
<td>7042 bytes (21)</td>
</tr>
<tr>
<td>18</td>
<td>4819 bytes (18)</td>
</tr>
<tr>
<td>15</td>
<td>2617 bytes (15)</td>
</tr>
<tr>
<td>12</td>
<td>2006 bytes (12)</td>
</tr>
<tr>
<td>9</td>
<td>1393 bytes (9)</td>
</tr>
<tr>
<td>6</td>
<td>793 bytes (6)</td>
</tr>
<tr>
<td>3</td>
<td>447 bytes (3)</td>
</tr>
<tr>
<td>1</td>
<td>227 bytes (1)</td>
</tr>
</tbody>
</table>
Results

- 500 ms to download and install plugins
- 22.3 us to scale individual packet
- 45,000 packets can be scaled per second
- 360 Mbit/s throughput (1KB packets)
- + 15 dB PSNR active vs non-active
Experiences and Reflections

• Us: Netbsd, Rest of world: Linux
• Mbufs, Gates, Flow Ids, Filters, etc.
  – lots of details
• Performance
• Security
  – Per packet vs. Per plugin
• Applications
  –