Predicting and Controlling Resource Usage in a Heterogeneous Active Network

w3.antd.nist.gov/active-nets
Outline of Presentation

- Motivations
- NIST solution to predict CPU requirements of an active packet on any node:
  - Models in brief
  - Prediction accuracy
- Application of NIST model to improve CPU-resource control in nodes
- Introduction to GE Active Virtual Network Management Prediction (AVNMP) a network load prediction system
- Enhancement of AVNMP by introduction of NIST models
- Future work
Motivations

Growing Population of Mobile Programs on Heterogeneous Platforms

SCRIPTING ENGINES & LANGUAGES

APPLETS & SERVLETS

dlls, dlls, and more dlls

MOBILE AGENTS

Active Networks
Active Networks Overview

**Principle:** Active packets carry not only data but also the code to process them which is executed at active nodes.

**Example:** An application that sends MPEG packets can specify an intelligent dropping algorithm to be applied at intermediate nodes if congestion is detected.

**Advantage:** Fast and easy deployment of customized network services.
Motivations

Sources of Variability in Active Packet Execution Time
Motivations

Threats and Needs

Without a means to express and predict CPU cycles needed to execute an active packet:

- Packets can consume excessive CPU time on a node or a set of nodes, causing denial of services to other packets
- A node can’t schedule its CPU resources to meet a packet’s performance requirements or other QoS requirements
- An active application can’t discover a route meeting its performance requirements
- Usage-based pricing simulations are impossible
NIST Model at a Glimpse

Monitor System Calls to NodeOS

Generate AA Execution Trace

Generate AA Model on node X

Node Model

Model of Node X:
- read: 40 cc
- kill: 18 cc
- EE: 13 cc

Model of Node Y:
- read: 20 cc
- kill: 45 cc
- EE: 9 cc

Scaling AA Model

Model of Node Y:
- read: 20 cc
- kill: 45 cc
- EE: 9 cc

Multiple Monte Carlo Simulations

AA CPU requirements prediction on node Y
# NIST Model Prediction Accuracy

<table>
<thead>
<tr>
<th>EE</th>
<th>AA</th>
<th>Node X</th>
<th>Node Y</th>
<th>Predictions after scaling with speed ratio</th>
<th>Predictions with NIST model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Error on mean prediction</td>
<td>Error on high percentiles prediction</td>
</tr>
<tr>
<td>ANTS</td>
<td>Ping</td>
<td>machine A</td>
<td>machine B</td>
<td>94</td>
<td>110</td>
</tr>
<tr>
<td></td>
<td></td>
<td>machine D</td>
<td>machine C</td>
<td>31</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td></td>
<td>machine E</td>
<td>machine C</td>
<td>23</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>Multicast</td>
<td>machine B</td>
<td>machine E</td>
<td>22</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>machine C</td>
<td>machine D</td>
<td>-11</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>machine A</td>
<td>machine C</td>
<td>226</td>
<td>209</td>
</tr>
<tr>
<td>Magician</td>
<td>SmartPing</td>
<td>machine E</td>
<td>machine C</td>
<td>34</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>machine B</td>
<td>machine C</td>
<td>121</td>
<td>103</td>
</tr>
<tr>
<td></td>
<td></td>
<td>machine A</td>
<td>machine D</td>
<td>287</td>
<td>281</td>
</tr>
<tr>
<td></td>
<td>SmartRoute</td>
<td>machine E</td>
<td>machine D</td>
<td>14</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>machine D</td>
<td>machine C</td>
<td>15</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td></td>
<td>machine C</td>
<td>machine A</td>
<td>-81</td>
<td>81</td>
</tr>
</tbody>
</table>
Improved CPU Usage Control

Control = Kill packets which execute above 99th percentile of active audio packet execution time

Real:

<table>
<thead>
<tr>
<th></th>
<th>Average execution time per packet:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(2278<em>M+455</em>8.29)/(2778+455)</td>
</tr>
<tr>
<td>8.29 ms</td>
<td>2,769,487 cc</td>
</tr>
<tr>
<td>4.76 ms</td>
<td>1,589,382 cc</td>
</tr>
<tr>
<td>23.99 ms</td>
<td>2,398,702 cc</td>
</tr>
</tbody>
</table>

Experiment #1: predictions based on execution time on sender and processor speed ratio

- 8.29 ms = 2,769,487 cc
- 8.29 ms = 829,187 cc
- Average execution time per packet:
  (2278*M+455*8.29)/(2778+455)
- 2186 good packets are killed

Experiment #2: predictions obtained with NIST model

- 4.76 ms
- Average execution time per packet:
  (2278*M+455*4.76)/(2778+455)
- Expected Improvement: 0.59 ms saved per packet
- Experimental Result: 0.63 ms saved per packet!
- 23.99 ms
- Only 19 good packets are killed
- Improvement = 2167 packets saved!
Improved Network Load Prediction

AVNMP in Brief

Overlay network simulates application traffic ahead in virtual time.

Experiment #1: CPU predictions based on average load on sender node and processor speed ratio
Experiment #2: CPU predictions obtained with NIST model
For both experiments: tolerance before rollback = 10%.
Improved Network Load Prediction

Experimental Results

<table>
<thead>
<tr>
<th></th>
<th>Exp#1: sender values scaled with processor speed ratio</th>
<th>Exp#2: CPU prediction with NIST model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>first intermediate node</td>
<td>second intermediate node</td>
</tr>
<tr>
<td>maximum look ahead (seconds)</td>
<td>-101</td>
<td>-20</td>
</tr>
<tr>
<td>Rollbacks</td>
<td>92</td>
<td>42</td>
</tr>
</tbody>
</table>

AVNMP improvement on the first intermediate node:
Future Work

Improve NIST models
- trace-based model has limitations that could be overcome
  with models that learn or with models that consider node-dependent conditions
- investigate prediction based on competition
- investigate alternate models: white-box model currently underway
- characterize error bounds

Improve AVNMP performance