Resource Management and Containment for Active Services

M. Ranganathan, Doug Montgomery, Kevin Mills
Advanced Networking Technologies Division
National Inst. Of Standards and Technology
Gaithersburg, MD
mranga@nist.gov
Observations

• Programmable Networks Trends:
  – Not in the data plane for IP.
  – More promising in the control plane.

• Custom call processing for internet telephony is a promising area:
  – People really want it.
  – Service platforms and soft switches are shipping.
  – There is a lot of activity in standards groups (IETF, JAVA community).
  – Same resource problems as traditional Active Nets
    • Security, Accounting and Resource Control
Project Goals

• Create a user-programmable service platform for extended SIP-enabled IP Telephony Services.

• Can lessons learned from Active Networks be applied to programmable SIP call processing environments to enable user programmability?
  – Analogous to Active Networks in several ways.
  – Programmable SIP Call processing enable user injection of service code.

• Key Issues:
  – Security
  – Load and resource control

• Starting Point:
  – DARPA funded NIST Active Nets project (Virgine Galtier, Kevin Mills et al.), Active Services work.
SIP and SIP services

- SIP is a HTTP-like signaling protocol for IP telephony and conferencing.
- A SIP service is an event triggered piece of code that runs on a SIP server.
- Event is generated by arrival of a message at a server or change in server state.
- Event can be:
  - low-level at the level of individual messages.
  - or semantic (at the level of a call).
SIP Network Components

SIP Proxy
- Proxies SIP Invites
- Host Services
- Handle Registration

Redirect Server
- Redirects Invites

User Agent Client
- Send out an invitation

User Agent Server
- Field the Invite
Service Architecture

- **Service Containers**
  - CPL
  - CGI
  - Servlets

- **Event / Service Manager Interface**
  - JAIN Event APIs
  - JAIN Service APIs

- **Soft Switch Operating System Layer**
  - Network Protocols
    - Network Device Drivers
  - OS Resource Management Services

- **Soft Switch Hardware**

- **Event / Service Management**
  - Session Initiation Protocol (SIP)
SIP Services (Service Creation)

- Examples of simple programmable services:
  - Call forwarding based on time of day / caller location.
  - Call redirection based on caller.

- Much industry activity:
  - SIP CGI / SIP CPL / SIP Servlets
  - JAIN-SIP/JAIN-SIP-LITE/JAIN SCE/SCML
  - Current schemes constrain programmability for user uploaded services (e.g., CPL).

- Our Goal:
  - Fully general User Programmable SIP Services.
  - Domains of applicability: SIP Servlets, Upload able test scripts for SIP test tool.
Driving Application:
NIST-SIP Test System

Test service
Proxy
NIST SIP
Responder
Responder UAS
Responder UAC

UAS

Customizable State Machine Template for Load Generation
Requirements for Up-Loadable Test Scripts

• Security: Need to protect the test server from unauthorized access to resources.
• Resource containment: Need to protect the server from denial of service attacks.
Restricting access to resources

• Use existing solutions:
  – Restrict class loading.
  – Access to all sensitive resources (such as files and network) will be via resource monitors.
  – Use Security policies to define capabilities for resource access.
    • Security Manager to restrict resource access.
    • Only wrapped classes are available to service scripts.
Controlling Resource Usage of a Running Script

• Two problems:
  – Admission control: Service platform should have an interface to query the incoming service script for what resources it needs.
  – Run-time control: Service platform should be able to abort execution for misbehaving service scripts.
Generating the Resource Signature

• Resource Signature
  – A function that represents a service that can be used to determine whether or not a service will run.
  – An incoming service script declares what resources it will need by its resource signature.
• Signature can be used by container for admission control and load control.
• Signature can be generated manually by user or generated with a signature generation tool.
• Signature generation tool will:
  – Generate a function that can be called by the container to query for required resources.
Structure of a Resource Signature

- Follow the approach developed in the previous work at NIST (Galtier, Mills et al.)
  - Application is represented by a finite state model with probabilistic transitions between states.
  - Previous project only considered CPU resources. We will extend this to include message traffic and other relevant resources.
  - Admission control of scripts is done by examining current system load and expected runtime load of the incoming script.
  - A malicious script can lie about its resource signature so we need runtime enforcement.
Controlling Resource Usage of a Running Script

• Use byte code rewriting technique:
  – Determine basic blocks.
  – Call back to resource checking hooks at the end of each basic block to see if allowance has been exhausted.
  – Exit the service script if allowance has been exhausted.
    • Portable metrics such as byte code allowances will be used for CPU time representation.
    • Message count and size will be used for network.
    • Coarse grained metrics such as object allocation rate and size will be used for memory.
**NIST Prior Work: Galtier, Mills et al.**

- **Monitor at System Calls in Active Node OS**
- **Generate Execution Trace**
- **Generate Active Application Model**

**Scenario A:**
- sequence = "read-write",
- probability = 2/5

**Scenario B:**
- sequence = "read-kill",
- probability = 3/5

**Distributions of CPU time in system calls**:

**Distributions of CPU time between system calls**:

**Trace is a series of system calls and transitions stamped with CPU time use**

**Scaling AA Models**

<table>
<thead>
<tr>
<th>EE</th>
<th>AA</th>
<th>Mean</th>
<th>Avg. High Per.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANTS</td>
<td>Ping</td>
<td>0.86</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td>Mcast</td>
<td>0.40</td>
<td>1.9</td>
</tr>
<tr>
<td>Magician</td>
<td>Ping</td>
<td>0.44</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>Route</td>
<td>0.73</td>
<td>13</td>
</tr>
</tbody>
</table>

**100 bins-20000 reps**

- **Model of node Y:**
  - read 20 cc
  - write 45 cc
  - user 9 cc

- **Model of node X:**
  - read 30 cc
  - user 10 cc
  - write 20 cc

- **AA model on node Y:**
  - read 30*20/40 = 15 cc
  - user 10*9/13 = 7 cc
  - write 20*45/18 = 50 cc

**Stats**

- **Ping**
  - Mean: 0.86
  - Avg. High Per.: 0.9

- **Mcast**
  - Mean: 0.40
  - Avg. High Per.: 1.9

- **Route**
  - Mean: 0.73
  - Avg. High Per.: 13
Related Work


**JSeal2**: Mobile agent system that uses byte code rewriting for runtime resource enforcement: [http://www.jseal2.com/](http://www.jseal2.com/)

**KaffeOS**: Process isolation and resource containment in JAVA. [http://www.cs.utah.edu/flux](http://www.cs.utah.edu/flux)

**DARWIN**: Resource management for Application Aware networks

Deployment

- Developed technology will be deployed in our web test system and made available on the ABONE for experimentation.
- Resource monitoring and enforcement framework for SIP Servlets will be proposed to the JAVA community for comment and possibly incorporated into the servlet spec.
- Developed code will be distributed as part of the NIST-SIP package.
  - Already a popular package for prototyping and development (1000s of downloads).
  - Implements JAIN SIP and will incorporate Servlets.
  - Test tool already developed.
Schedules

- **Jan 2002:**
  - Exploration and evolution of the design.
- **August 2002:**
  - SIP Servlet implementation and development of resource monitor technology.
  - Release SIP Servlets as part of NIST-SIP 1.2
- **December 2002:**
  - Integration of resource monitor with the servlet engine.
  - Release SIP Servlets with resource control as part of NIST-SIP 1.3
Schedules

• **August 2003**
  – Integration into our test system
  – Gather more feedback and debug

• **December 2003**
  – Project completion and deployment on the ABONE.