Janos: A Java-Oriented OS for Active Network Nodes

Patrick Tullmann, Mike Hibler, and Jay Lepreau
IEEE JSAC, March 2001
Outline

- Introduction
- Goals
- Design
  - Moab
  - JanosVM
  - ANTSR
- Results
- Discussion
- Conclusion
Introduction (1/2)

- Java-oriented **Active Network Operating System**
- Emphasis on resource management and control of untrusted active applications written in Java
- Active node’s software architecture:
  - NodeOS
  - Execution Environment (EE)
  - Active Application (AA) layer
Goals (1/2)

• Untrusted code support
  – Execution of untrusted java code carried by the users’ packets

• Resource management
  – Memory
  – CPU
  – Outgoing network bandwidth
Goals (2/2)

- Performance
- Separable components
  - Not only make the components optimized to work together, but also useful independently
Design: Moab (1/3)

- NodeOS built upon the OSKit
- OSKit is a collection of device driver, POSIX APIs, filesystems, and network protocols
- POSIX (Portable Operating System Interface for UNIX)
- POSIX-reliant systems can be migrated to Moab easily
Design: Moab (2/3)

- Host for a single trusted EE
- Resources are specified precisely in term of the local node’s hardware
- Allow EE to perform memory management
- A packet buffer is used for storing incoming packets
Design: Moab (3/3)

• User can manage CPU usage within a domain
• A domain, similar to a process in a traditional OS, is the unit of resource control
Design: JanosVM (1/3)

- A virtual machine that accepts Java bytecodes and execute them on Moab
- Provides access to the underlying NodeOS interface through the Janos Java NodeOS bindings
- Based on KaffeOS, a JVM that provides the ability to isolate apps from each other and to limit their resource consumption
• Supports multiple, separate heaps, separate garbage collection threads for each heap, per-heap memory limits

• Strict separation of domains, each domain runs in its own namespace and in its own heap
Design: JanosVM (3/3)

- Provide a library for mapping platform independent resource specification into Moab’s hardware-specific specifications
Design: ANTSR

• Java runtime library based on ANTS 1.1
• Provides interfaces for untrusted, potentially malicious, AAs to interact with the system
• Hides critical JanosVM interface from the AAs
• Specifies per-domain resource limits
Results (1/2)

<table>
<thead>
<tr>
<th>Forwarding Path</th>
<th>Rate (Kpps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSKit</td>
<td>75.7</td>
</tr>
<tr>
<td>Moab cut-channel</td>
<td>48.7</td>
</tr>
<tr>
<td>C-based EE on Moab</td>
<td>45.0</td>
</tr>
<tr>
<td>Java-based EE on Moab</td>
<td>19.5</td>
</tr>
</tbody>
</table>
### TABLE II

**Relative Output Bandwidth Usage for Resource Limited Out-Channels Trying to Independently Saturate a Single Physical Link. The Measured Throughput Is Reported in Bytes Per Second.**

<table>
<thead>
<tr>
<th>Share</th>
<th>Measured throughput (Bps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2,053,608</td>
</tr>
<tr>
<td>2</td>
<td>4,094,836</td>
</tr>
<tr>
<td>3</td>
<td>6,145,442</td>
</tr>
</tbody>
</table>
Discussion

• Java’s performance is not good enough
  – JIT (Just-in-time) compilation can be used
  – Cross-platform
• Real-time traffic is not a concern in Janos yet
• Multitasking issue
Conclusion

• To provide comprehensive, precise resource control over the execution of untrusted Java bytecode
• To provide services and features in the appropriate layer, without overlap