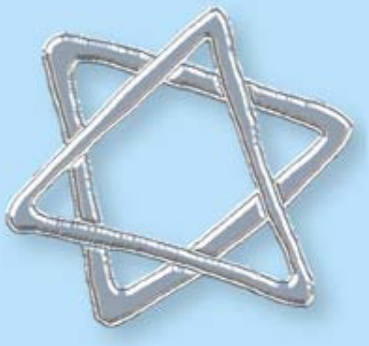


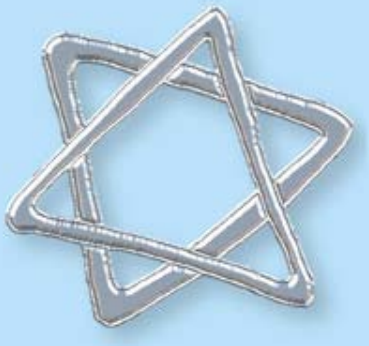
Janos: A Java-Oriented OS for Active Network Nodes

Patrick Tullmann, Mike Hibler, and Jay Lepreau

IEEE JSAC, March 2001

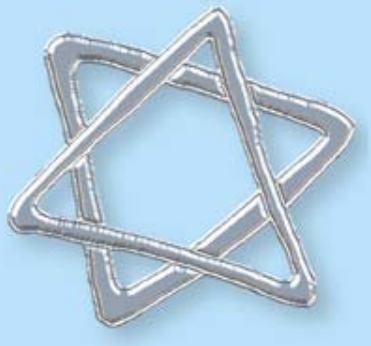
徐延源

2002/10/24



Outline

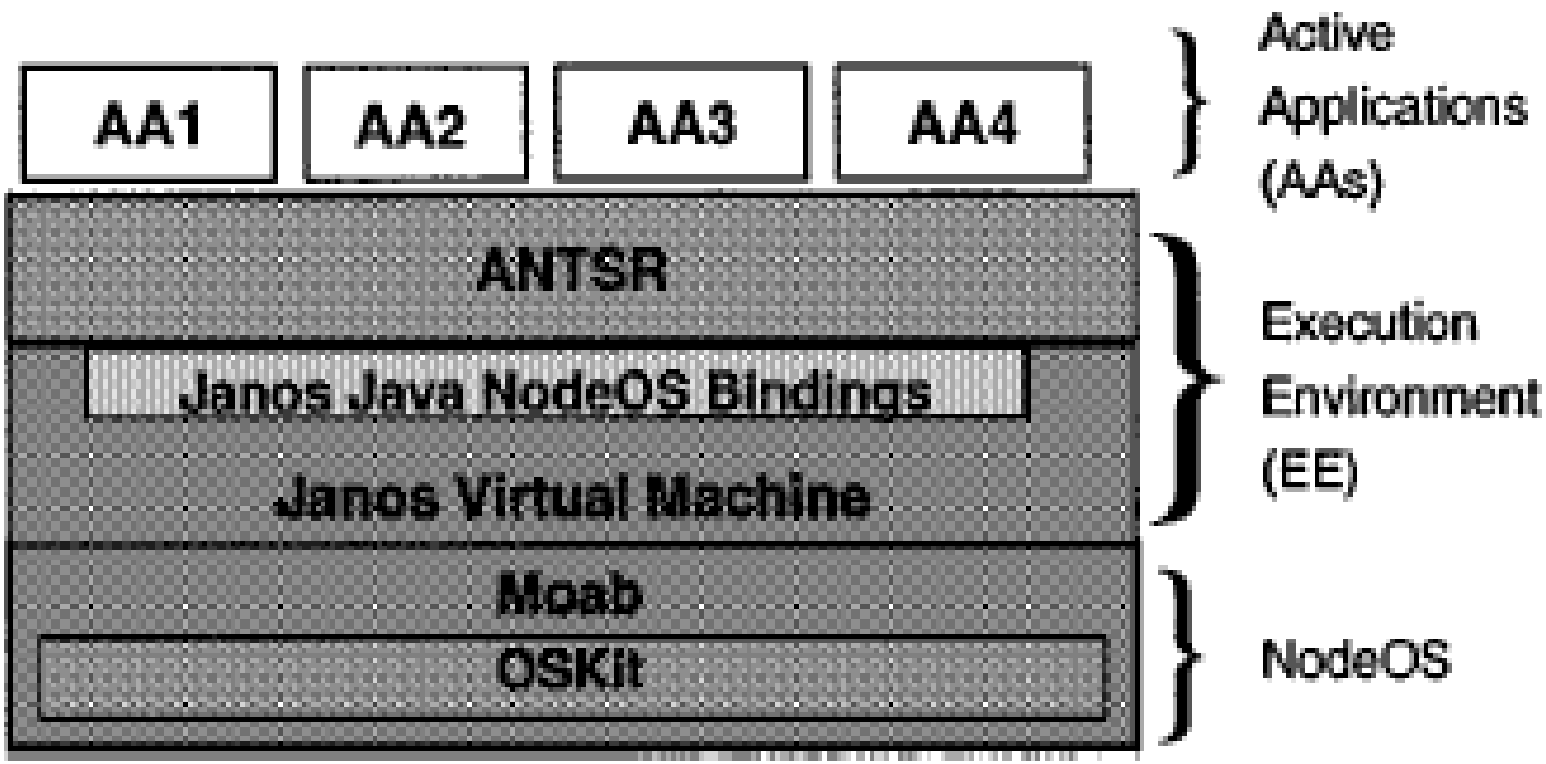
- Introduction
- Goals
- Design
  - Moab
  - JanosVM
  - ANTSR
- Results
- Discussion
- Conclusion

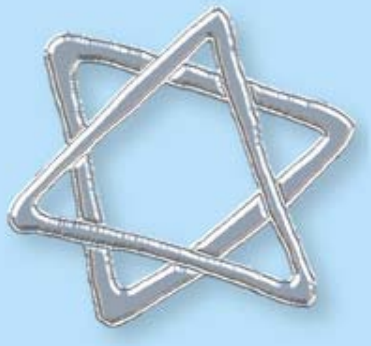


Introduction (1/2)

- 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

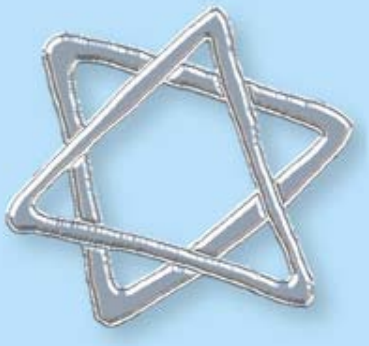
Introduction (2/2)





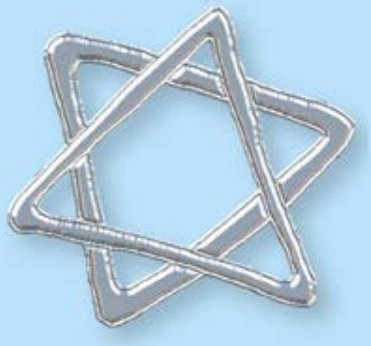
Goals (1/2)

- Untrusted code support
  - Execution of untrusted java code carried by the users' packets
- Resource management
  - Memory
  - CPU
  - Outgoing network bandwidth



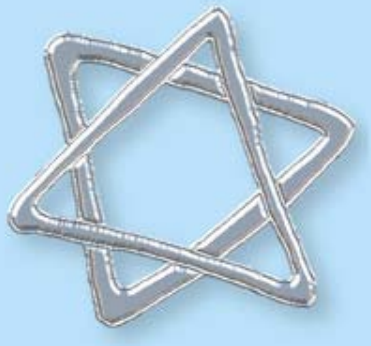
Goals (2/2)

- Performance
- Separable components
  - Not only make the components optimized to work together, but also useful independently



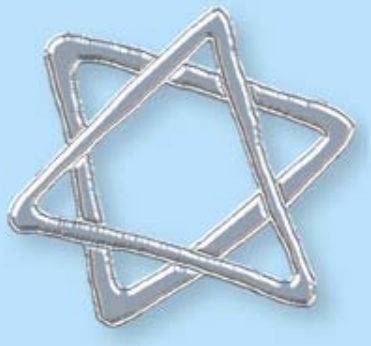
Design: Moab (1/3)

- 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



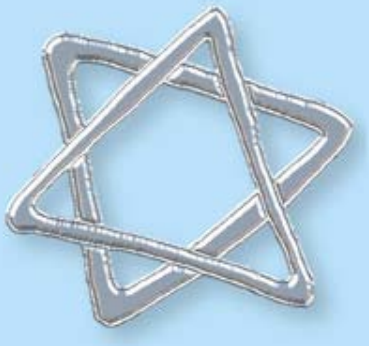
Design: Moab (2/3)

- 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



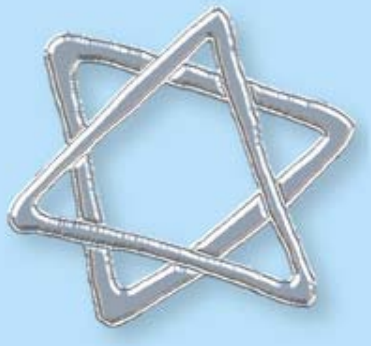
Design: Moab (3/3)

- User can manage CPU usage within a domain
- A domain, similar to a process in a traditional OS, is the unit of resource control



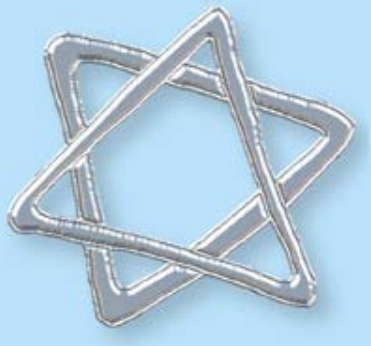
Design: JanosVM (1/3)

- 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



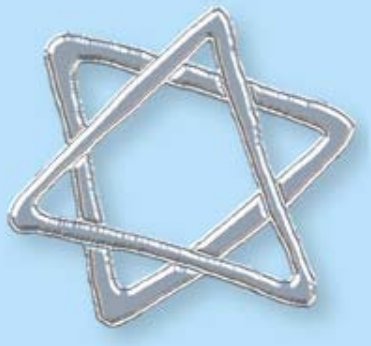
Design: JanosVM (2/3)

- 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



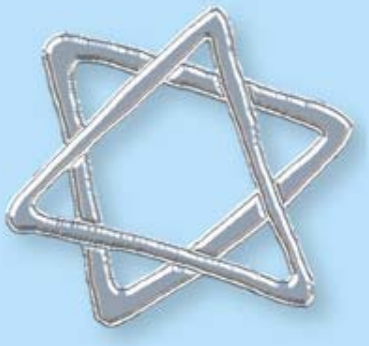
Design: JanosVM (3/3)

- 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)

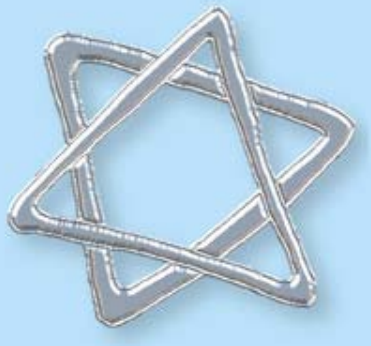
TABLE I
PACKET FORWARDING RATES AT VARIOUS LEVELS IN MOAB, IN THOUSANDS OF PACKETS PER SECOND.

Forwarding Path	Rate (Kpps)
OSKit	75.7
Moab cut-channel	48.7
C-based EE on Moab	45.0
Java-based EE on Moab	19.5

Results (2/2)

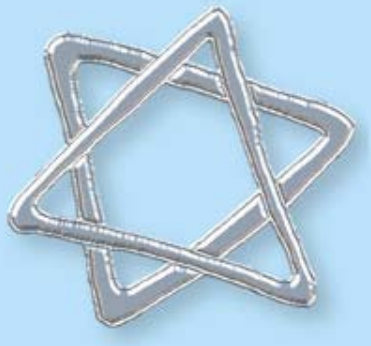
TABLE II
RELATIVE OUTPUT BANDWIDTH USAGE FOR RESOURCE LIMITED OUT-CHANNELS TRYING TO INDEPENDENTLY SATURATE A SINGLE PHYSICAL LINK. THE MEASURED THROUGHPUT IS REPORTED IN BYTES PER SECOND.

Share	Measured throughput (Bps)
1	2,053,608
2	4,094,836
3	6,145,442



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, presice resource control over the execution of untrusted Java bytecode
- To provide services and features in the appropiate layer, without overlap