Bees: A Secure, Resource-Controlled, Java-Based Execution Environment

Tim Stack Eric Eide Jay Lepreau
University of Utah
April 5, 2003

What is Bees?

- **Mobile code system that is**
  - **Realistically deployable because it addresses needs of node administrators**
  - **Realistically usable because it provides rich interface needed by service authors**
- **We believe may be the first such environment**

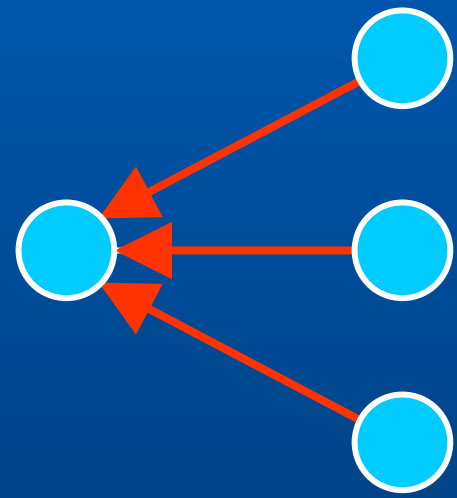
Key Features

- Flexible security primitives
- Resource control
- Flexible protocol composition
- Flexible control of packet propagation
- Isolates interaction with end-user apps

Bees integrates them all

A Motivating Application

- Motherboard sensor monitor
  - Spreads over network
  - Reports to server
  - Shuts down faulty nodes
- Ideal for active protocol
  - Flexible access to sensors
  - Not speed-critical



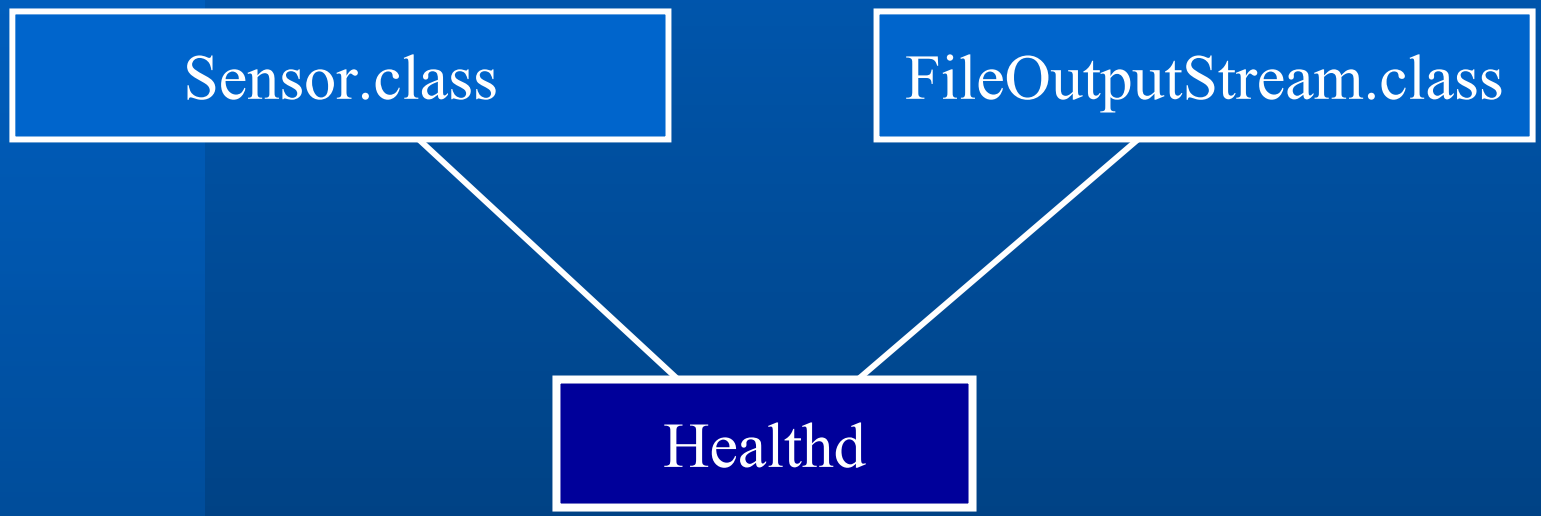
Health Reports

Node

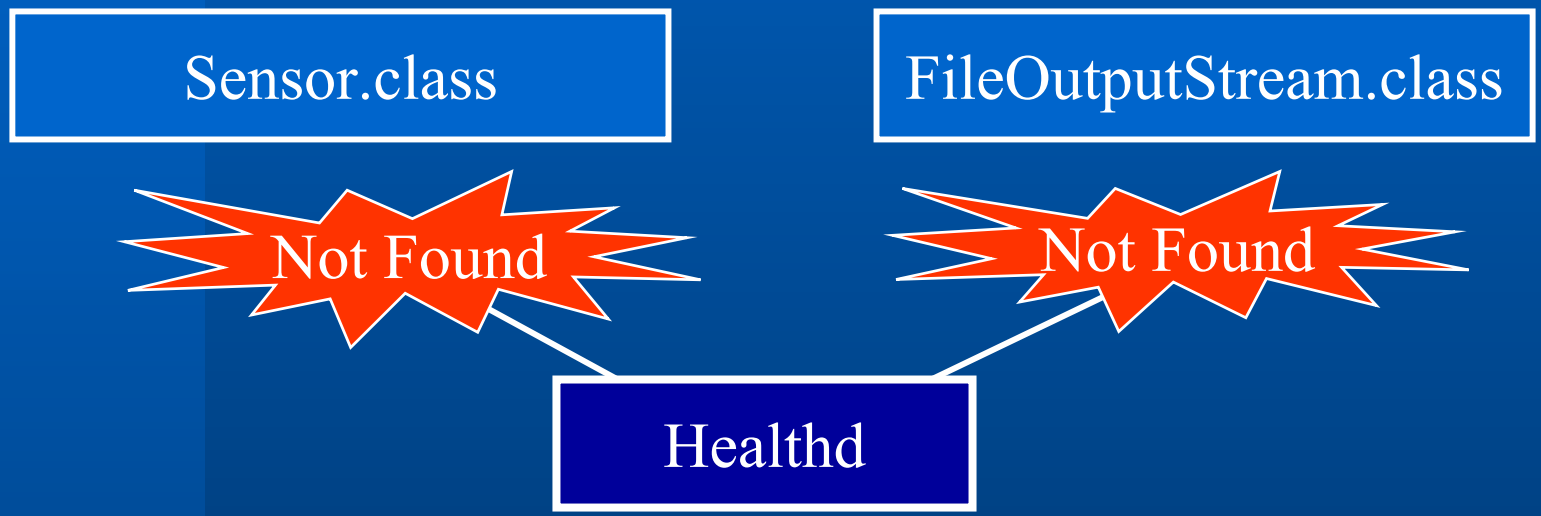
ANTS: Implementation

- Capsule
  - Packet associated with Java class through MD5 hash
- Protocol
  - Collection of capsule classes
- Application
  - Includes copy of protocol
  - Source of all capsules

ANTS: Security



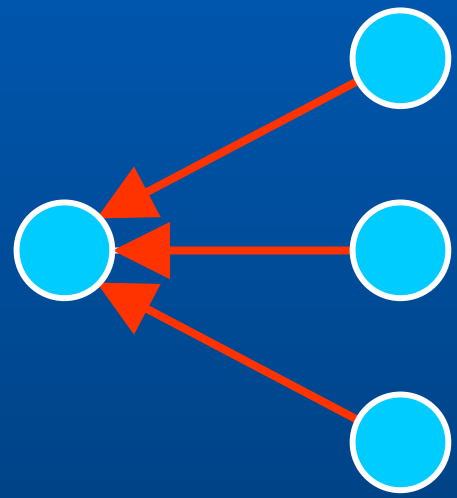
ANTS: Security



- **No security infrastructure**
  - **Can't read sensors**
  - **Can't log to file**

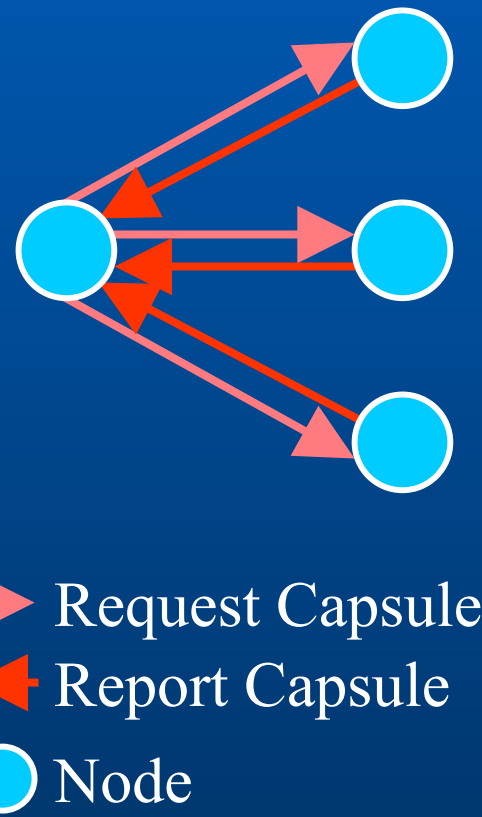
ANTS: Resource Control

- TTL controls resources
- TTL must be replenished



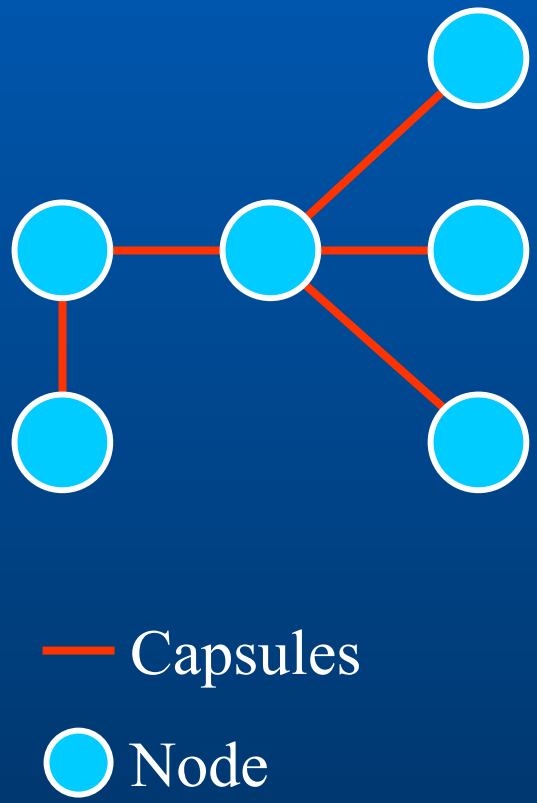
ANTS: Resource Control

- TTL controls resources
- TTL must be replenished
  - Server sends requests
- Problems
  - More network traffic
  - Topology not discovered



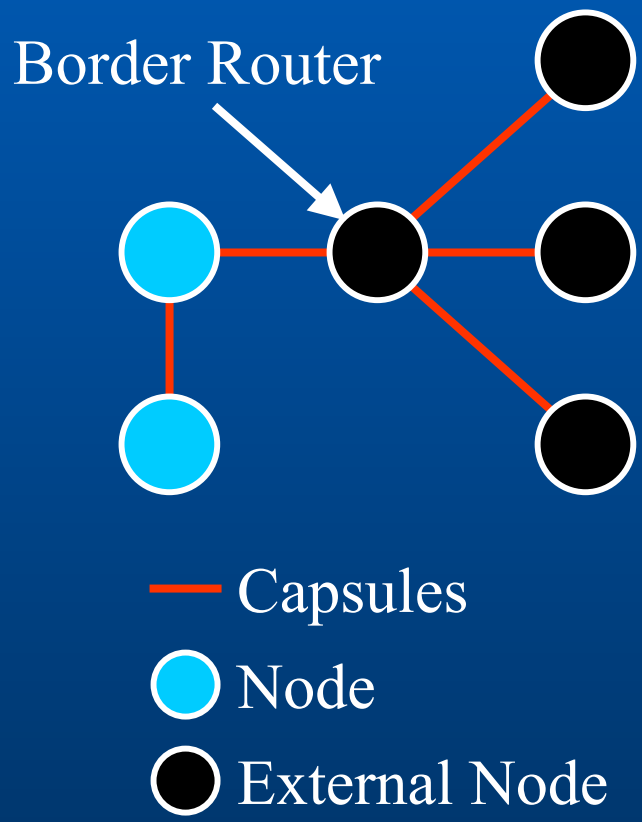
ANTS: Node Discovery

- **Discover topology**
  - **Just send to neighbors**



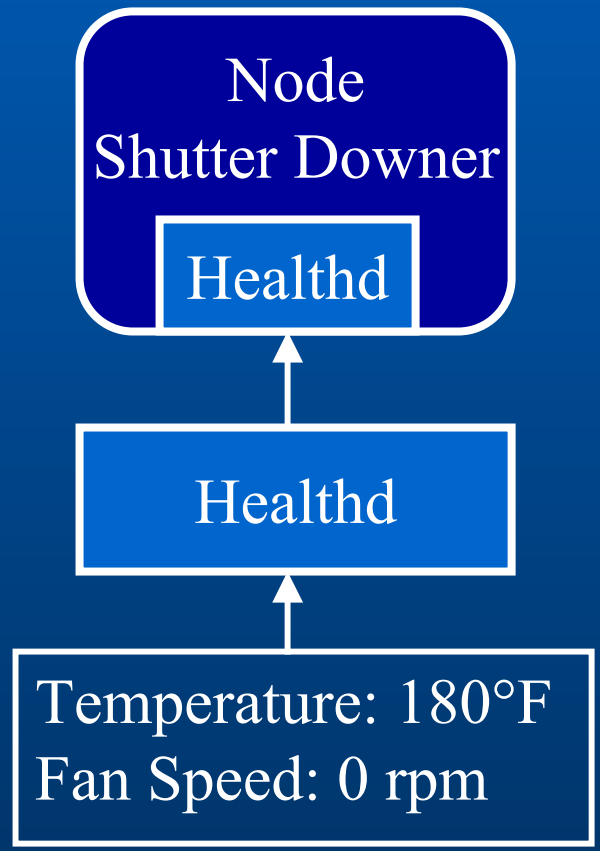
ANTS: Node Discovery

- **Discover topology**
  - **Just send to neighbors**
- **Problems**
  - **Protocol containment**
  - **More TTL issues**
  - **Hard to reuse**



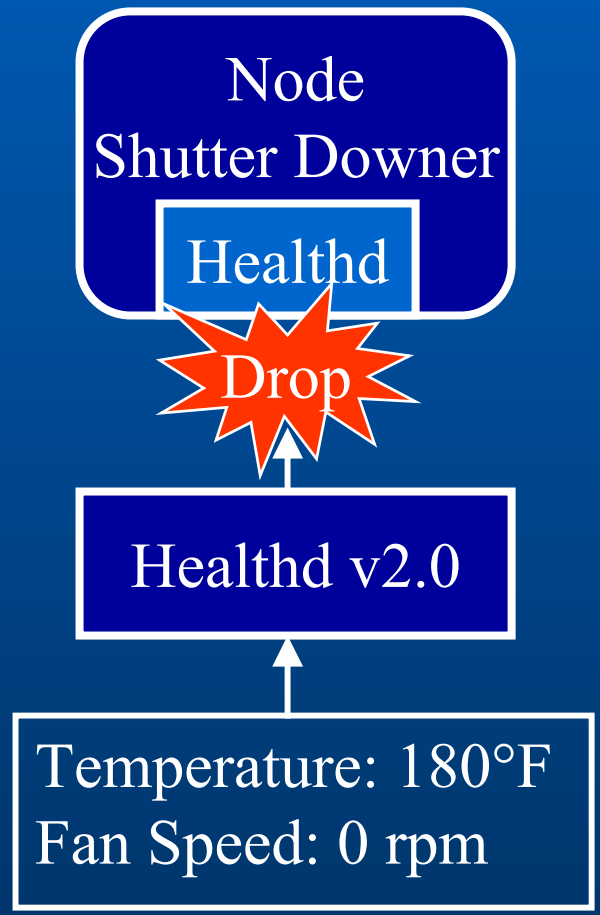
ANTS: Endpoint

- **Node unhealthy**
  - **No shutdown permission**
  - **Tell application**



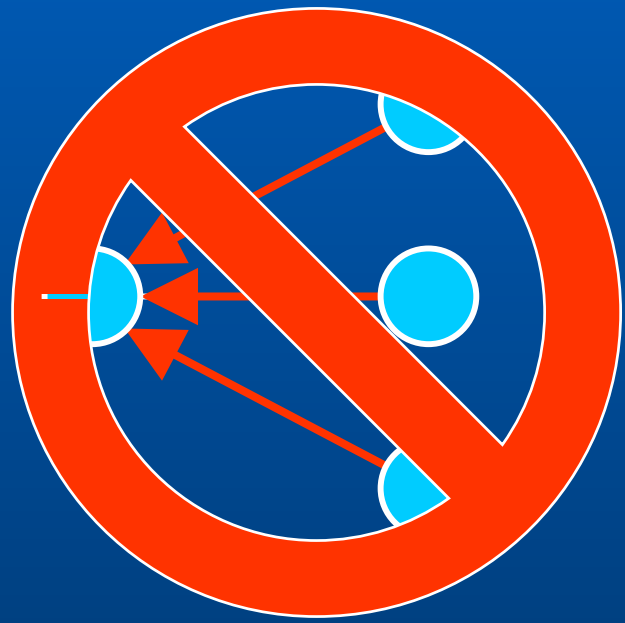
ANTS: Endpoint

- **Node unhealthy**
  - **No shutdown permission**
  - **Tell application**
- **Version change**
  - **Capsule hash mismatch**
  - **Application must be updated manually**



ANTS: Assessment

- Reality intervenes
- What is wrong?
- Wrong type of EE
- Richer EE needed



Lean vs. Rich

Lean

- Little to no state
- Forwarding loop only
- Specialized language
- Simple resource control/accounting

- Example SNAP/ANTS

Rich

- Node resident state
- Threads, timer events
- General language
- Complex resource control/accounting

- Example: Bees

Overview

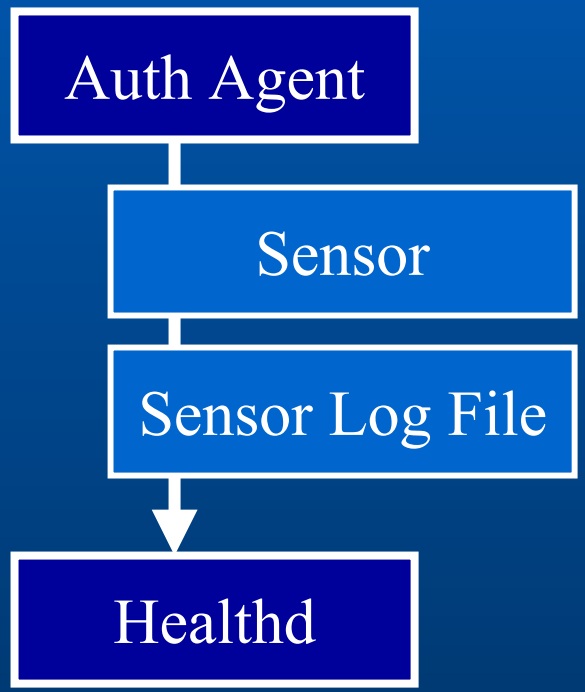
- Bees
  - Security
  - Resource control
  - Protocol composition
  - Application interaction
  - Details of code migration
- Related work
- Conclusion

Security: Isolation

- **Multi-process JVM**
  - **Isolates active code**
  - **Process holds state, privileges**
- **Process is unit of resource control**
- ***Auth Agent*** **creates and terminates**

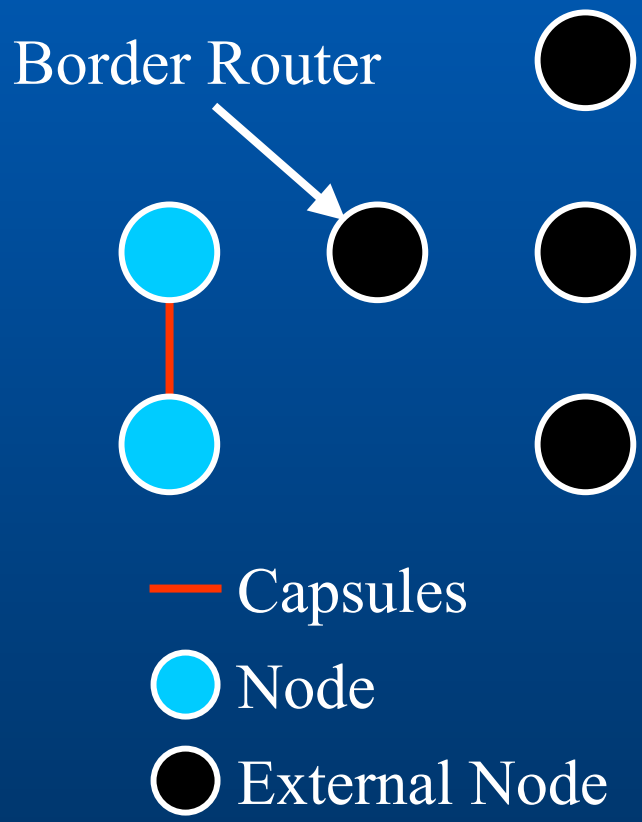
Security: Capabilities

- Capability-based security mechanism
- Examples
  - Files
  - Cryptographic keys
  - Neighbors
- Distributed by Auth Agent



Example: Node Discovery

- **Border neighbor withheld**
- **Privileges needed to escape**



Resource Control

- Janos infrastructure
  - CPU, network, and memory
- Process is unit of control
- Termination reclaims resources
- Network controls
  - Bandwidth limits not enough
  - TTL too restrictive

Network Control

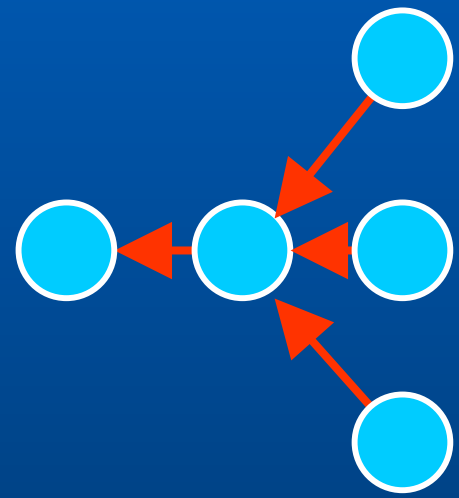
- Allow only solicited forwarding
- External stimuli
  - Timer, capsule receipt, application, …
- Fine grained operations
  - Forward to neighbor
  - Return to source
  - Multicast to neighborhood
  - Transform to another capsule type

Capsule Operations

- Capsule operation counters
  - Protocol author defines initial values
  - Stimuli replenishes values
  - Decremented on use
  - Operations disallowed when zero
- Initial values limited by Auth Agent

Example: Resource Control

- **Report capsule**
  - **Replenished by timer**
  - **Sent**
  - **Further use stopped**
- **Forwarding is similar**
  - **Replenished by receive**



Protocol Composition

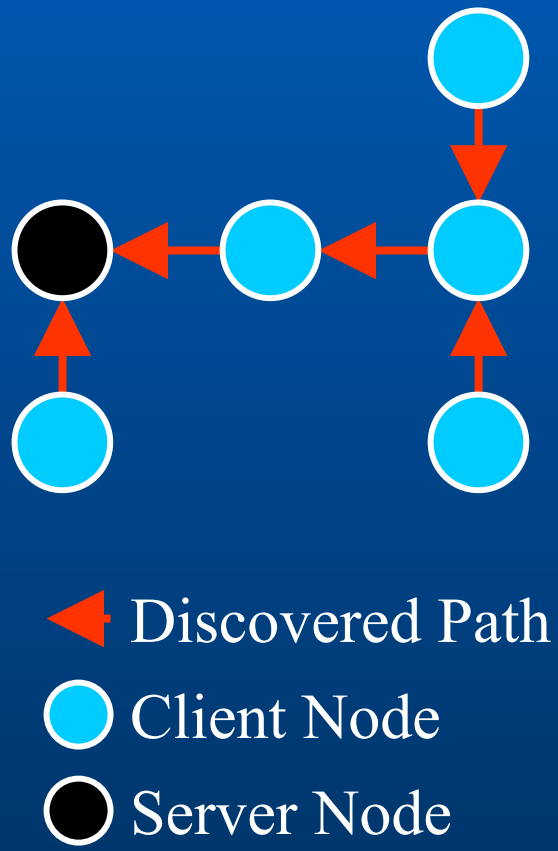
- **No protocol is an island**
  - **Protocols depend on each other**
- **Protocol is the unit of composition**
  - **Primary paired with *companions***
- **Protocols form a hierarchy**
- **System provided**
  - **Code downloader**

Pathfinder

- Primitive routing protocol
- Routing scenarios:
  - Client to server
  - Server to all clients
  - Server response to client request
- Implementation
  - Spanning tree behavior
  - No addresses

Example: Node Discovery

- **Periodic broadcast**
  - **Finds path to server**
  - **Spreads code**

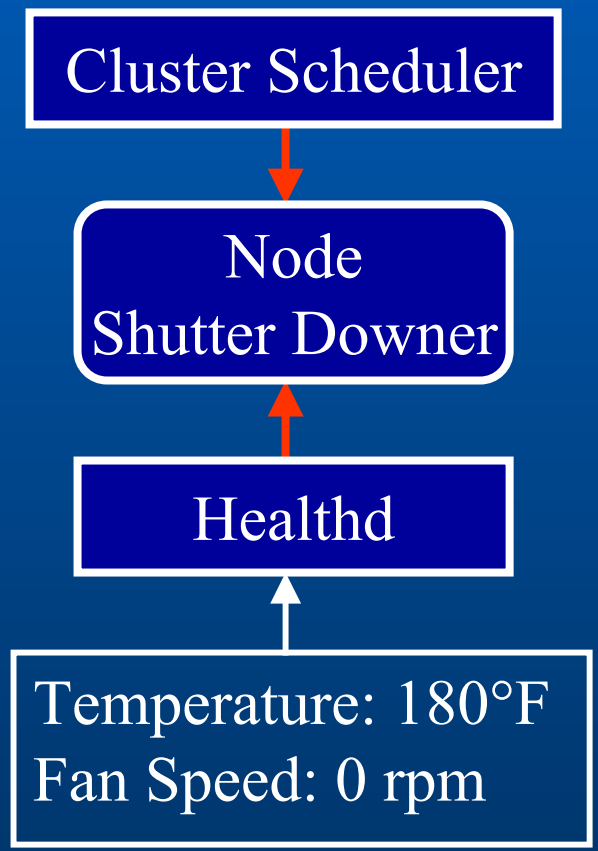


Application Interaction

- *Protocol Session* provides application interface
- Trust barrier
  - Only byte arrays are exchanged
- Abstracts raw protocol
  - Insulation from versioning issues
- Similar to standard socket interfaces

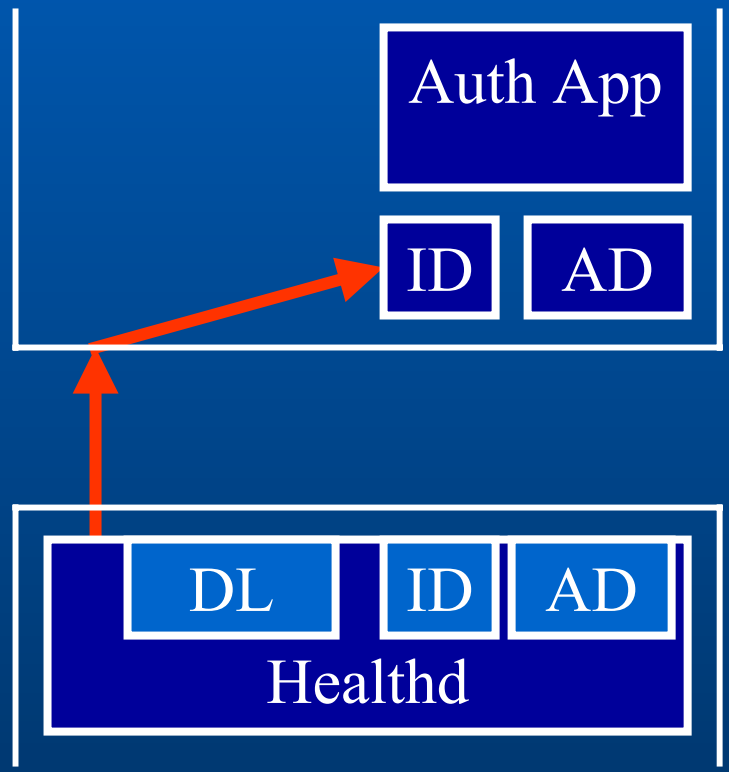
Example: Endpoint

- **Node unhealthy**
  - **Tell application**
- **Other protocols can use same interface**



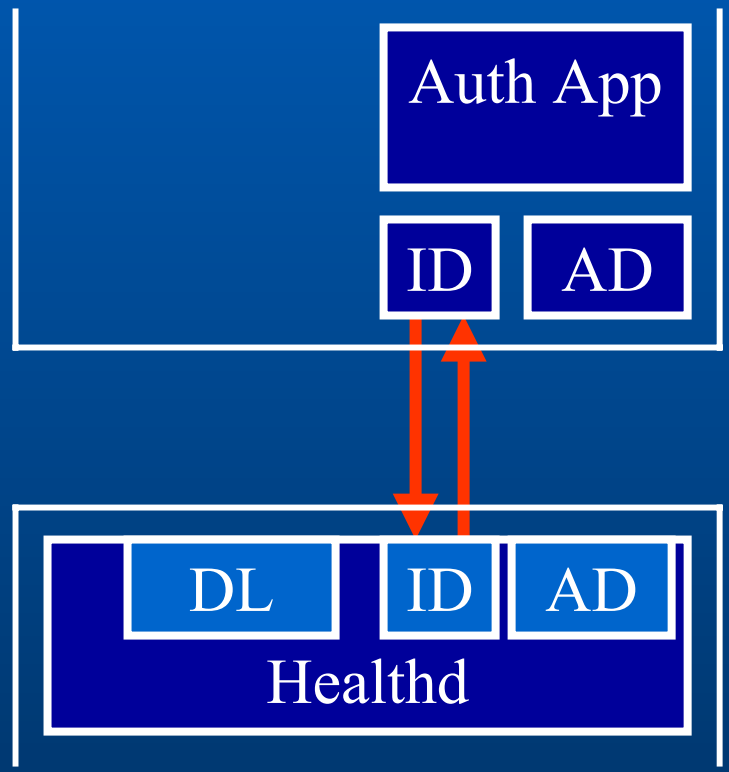
Code Migration

- **Unknown capsule**



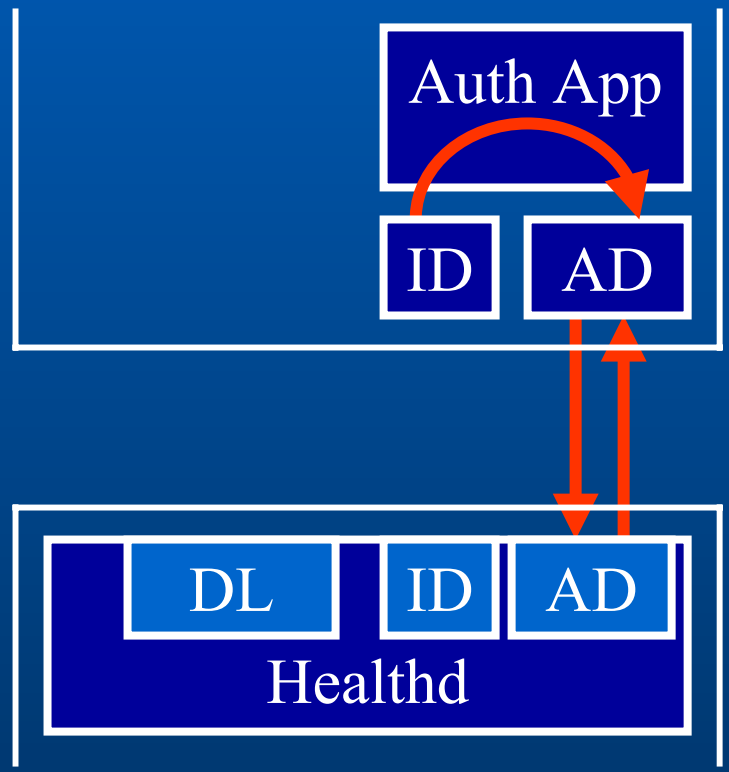
Code Migration

- Unknown capsule
- Map capsule to Healthd



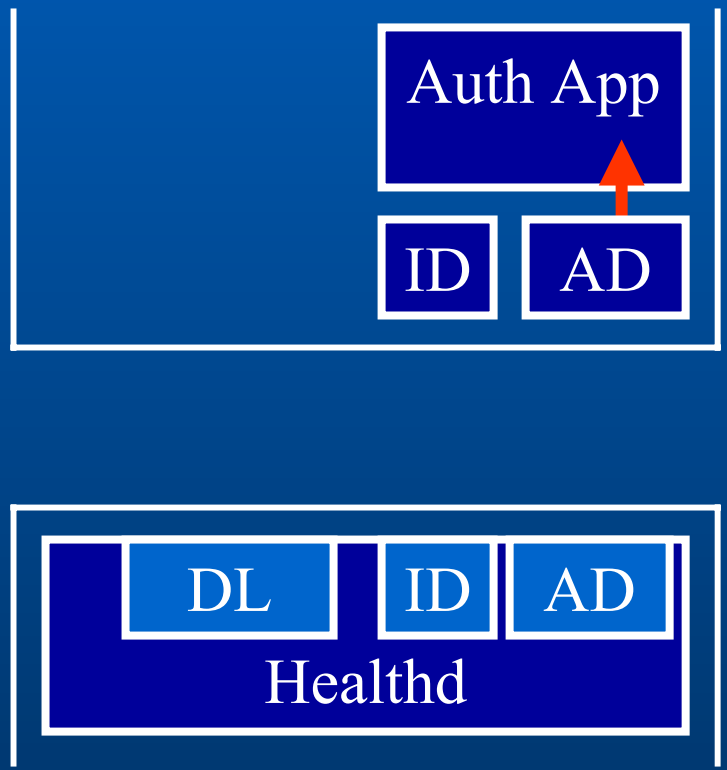
Code Migration

- Unknown capsule
- Map capsule to Healthd
- Download auth data



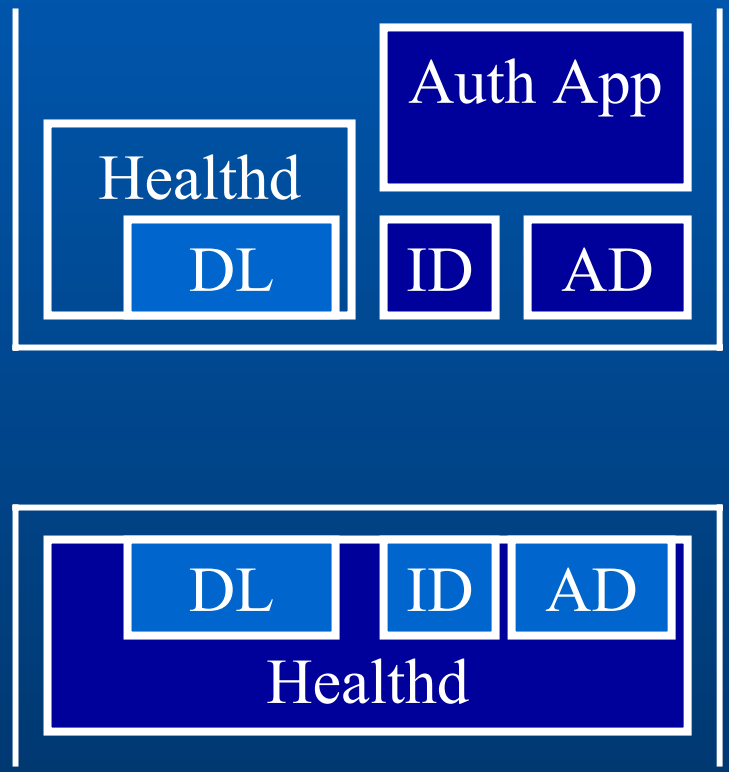
Code Migration

- Unknown capsule
- Map capsule to Healthd
- Download auth data
- Check auth data



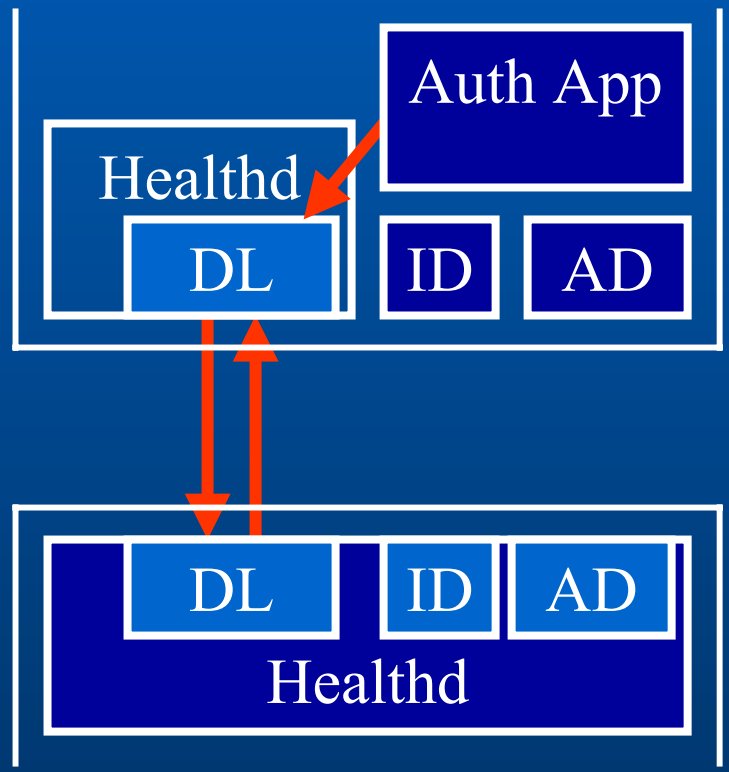
Code Migration

- Unknown capsule
- Map capsule to Healthd
- Download auth data
- Check auth data
- Create process



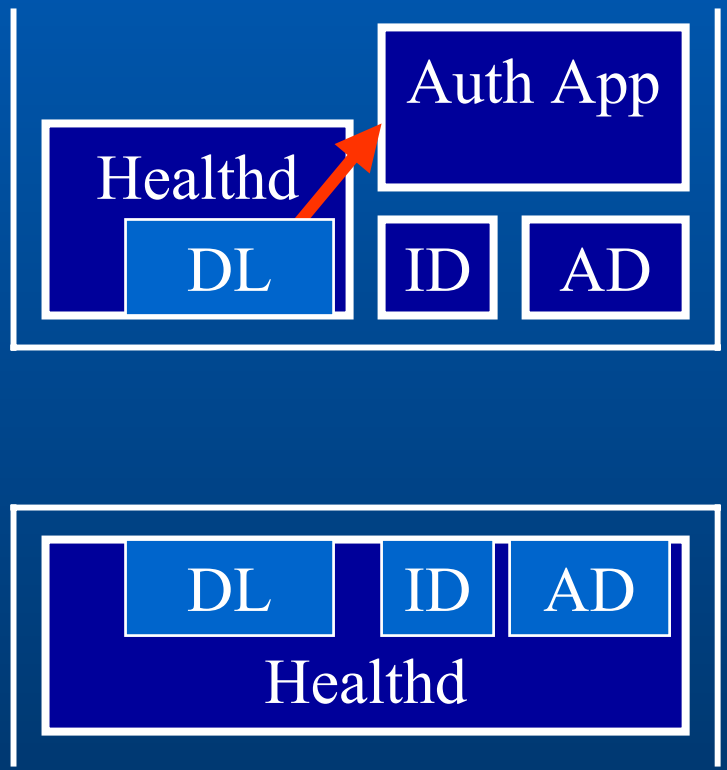
Code Migration

- Unknown capsule
- Map capsule to Healthd
- Download auth data
- Check auth data
- Create process
- Start download



Code Migration

- Unknown capsule
- Map capsule to Healthd
- Download auth data
- Check auth data
- Create process
- Start download
- Finish download



Related Work

- Resource control
  - RCANE[Menage00], SNAP[Moore01]
- Security
  - SANE[Alexander98], SANTS[Murphy01]
- Protocol composition
  - CANES[Bhattacharjee99]

Bees v0.5.0

- 50,000+ Lines of Code
- 30-page manual
- Example application
- Available at:

  www.cs.utah.edu/flux/janos

Conclusion

- Rich environment
  - Support for node administrators
  - Support for protocol authors
- Key Features
  - Security and resource control
  - Protocol composition
  - Isolates interaction with end-user apps