The NESTOR Project
Automating Configuration Mgmt

Alexander V. Konstantinou
Yechiam Yemini
Distributed Computing & Communications Laboratory
Columbia University

Self-Organizing Networks

- Self-organizing = adapt to changes
- ANets are self-organizing: change is the only constant
- Adaptation requires independent mechanism
- NESTOR provides self-organizing capabilities to networks
  - Maintains a model of network: objects-relationships
  - Detects changes
  - Adapts to changes by propagation among related objects
  - Controls propagation through constraints

Results

- Technology Results
  - NESTOR core technologies:
    - Unified data & semantic model for self-configuring networks
    - Programmable change policies: change propagation + constraints
    - Architecture
      - Network Management apps: enable mobile users
      - Security apps: maintain security through changes in use
  - Impact
    - Telcordia Technologies: smart firewalls
    - ANET Demos: UCLA/Utah/UCB
    - ABONE [soon]

NESTOR Architecture & Operations

- NESTOR architecture:
  - Resource Repository
  - Constraint & Propagation Manager
  - Active Application Manager
  - Resource Repository

Modeling the World

- Adapters poll devices/services
- Changes reflected as model updates
- Constraints & propagation rules trigger before model commit

Modeling the World (2)

- Real world issues:
  - No isolation
  - Compromising transactions
  - Management application starvation (priority issue)
- Assumptions on agent behavior required
NESTOR Data Modeling

- Unified object-relation data model
  - Classes model static configuration (specialization through inheritance)
  - Relations model dynamic deployment
  - Challenge: defining unified schema

Does not express semantic information
- E.g., interface IP address must match network mask
- E.g., interface MTU is should match the router’s MTU

User Interface & API

NESTOR Constraints & Propagation

- Constraints on valid configuration (declarative)
  - Example: IP interface netmask must match address
    ```java
    IpInterface.allInstances
    ->select(I | (I.address != null) and
    (I.netmask != null))
    ->forAll(I | I.netmask.matches(I.address))
    ```

- Configuration propagation rules (operational)
  - Example: Video active app. packet size <= interface MTU
    ```java
    MyVideoAA.allInstances
    ->forAll(app | app.packetSize :=
    app.connectedVia->min(link | link.mtu))
    ```

Constraint & Propagation Challenges

- Simple navigation of relationships
- Propagation cycles
  - Change propagates over relations
  - Static analysis may be too conservative
- Bounding propagation
- Distribution of computation

NESTOR-based AN Management

- Local NESTOR repository on each node
- Standard (simple) models
- NodeOS, EEs, AAs discover local repository
  - Register objects, constraints, and propagation rules
- Query repository to discover & configure needed services
- Relations between AAs span repositories
- Unified instrumentation for security and accounting

NESTOR @ ABONE

- Anetd Adapter
  - Read/write configuration instrumentation
- Virtual “live” configuration browsing
- Public audit of configuration changes
- Virtual Active Networks (VANs)
- Columbia 12-node VAN/ABONE test-bed
- EE/AA Author Instrumentation Kit
  - Load/unload/monitor EEs
  - Discover system configuration
  - Export instrumentation/constraints

Resource Repository
Abone host
Web browser
(browse/audit)
Telcordia Smart Firewalls

 difficulties include:
- Manual configuration requiring security expertise
- Networks are too dynamic
- Current configuration tools cannot validate
- Security policies must be enforced across multiple administrative domains

Example: can someone telnet into network?

Telcordia Smart Firewalls (2)

Security policies:
- High level goals (allow/deny)
- Invariants that must hold (not conditions-actions)

Validation and Secure Change Management:
- Policy engine validates entire network configuration
- Supports what-if queries
- Automatic policy enforcement using NESTOR
  - Network discovery/update, transactional commit

Centralized user interface for network security administration

NESTOR Security

NESTOR provides security perimeter for management
- Repository user authentication (X.509)
- Secure communications (TLS)
- Object-attribute level ACLs
- Adapters trust repository
  - Real world treated as a user (with associated permissions)
  - Propagation path analysis
    - Modeling (unified instrumentation) supports domain analysis

API Summary

- Session
  - Repository discovery
- NESTOR Repository
  - Create transaction
  - Create object
  - Lookup objects (by class/attribute)
  - Subscribe for changes
- Standard Model
  - Link, Network, Application layer objects
- Agent utilities
  - Morphing and polling

Summary of Results

Prototype implementation:
- Java/Uni based (>100K lines)
- Distributed object-relation repository + standard API + standard model
- Model compiler & constraint/propagation interpreter
- Adapters: Linux, Cisco IOS, SNMP, VAN, ANetD
- Browser: repository, performance & topology visualization
- Packaged & stable

Demonstrations:
- Telcordia demonstrations

Technology Transfer:
- Telcordia Technologies: DARPA distributed firewall project
- UCLA/UCB/Utah: DARPA Active Network integration demo
-Soon: ABONE deployment

Current Research & Plans

Security features
- Propagation path analysis
  - Formal propagation model
  - Propagation domain analysis
- Public ABONE deployment
- Operational configuration recovery
- Auditing