Policy Validation in Policy-Based Networks

Implementation of Various Reasoning Engines

Presenter: Kamran Rajput
Presentation Layout

- Introduction
- Existing Solutions
- Policy Languages
- Reasoning Engines
- Implementation
- Discussion/Questions
The IETF Architecture

- High-Level Business Policies
- Device-Independent Policies
- Device-Dependent Policies
- Policy Management Tool
- Policy Decision Point (PDP)
- Policy Enforcement Point (PEP)
- Policy Repository
Proposed Architecture (ATNAC’04)
If <condition> then <action>

- Policy constructs define functionality required to perform desired traffic conditioning.
  - VoIP: Minimal loss, delay and jitter
  - Web Buying Transaction: Specific Encryption Algorithm, Response time

- QPIM provides specific classes to enable DiffServ and IntServ conditioning to be modeled

- The QPIM class definitions create instances of various policy constructs such as QoS actions and conditions.
  - Actions include rate limit, bandwidth allocation and jitter control
  - Conditions can select traffic conditioning according to a complex Boolean expression.

- **Policy Continuum - a mapping of mechanism from one layer to another**
Weaknesses: The IETF Policy Continuum

- The IETF’s Rule-based approach is a general guideline of the process
  - No workable details about implementation (any language, any formal notation or other technological standard)
  - Easy to specify, Difficult to implement sophisticated validation checks (syntactic, semantic)
  - No standardization in terms of policy specification language/technology

- No Evidence of any policy validation work or process

- Need for Policy specification standards on how to exchange information between entities/domains
  - Policy negotiation and merging between two different domains

- Interoperability requires agreement on a common schema between two different domains
  - no evidence of defined schema for QoS in the IETF models.
Existing Approaches

• Policy Languages
  – PONDER (OO, declarative)
  – KAoS (DAML+OIL ontologies)
  – Rei (deontic concept-based)

• XML-Based policy Languages
  – RuleML, WSPL

• Logic programming Languages
  – Prolog, Logikus, OCL
Problem

• “The definition of policy conflict depends upon the approach used to specify policies” (Verma)
• “The representation chosen to describe policies and their context determines the flexibility, extensibility, and amenability to analysis of a given implementation” (KAoS)
• Proprietary
• Domain or application specific
• Difficult to integrate with other domains
XML: Rules Specification

- XML: standard for sharing and exchange of data on the web
  - Text-based and platform independent
- Isolating rules from application code allows changing rules without software modification
  - Adaptability, Flexibility and Efficiency

How can we achieve these benefits in Networking domain?
RuleML (Rule Markup Language)

- Allows expression of rules in a Modular way
- Uses distinct and standard XML tags
- Rule base composed of facts and rules
- In-Direct Execution of Rules
- Rules translated to any Inference engine language (Jess, ABLE, Prolog)
- Independent of reasoning engine, allows exchange of rules between different engines
Web Services Policy Languages (WSPL)

- Developed at Sun Microsystems
- Strict subset of OASIS eXtensible Access Control Markup Language (XACML)
- Supports wide range of policy specifications (QoS, authorization, reliable messaging, privacy)
- Supports merging of Policies for Policy negotiation
  Policies can be compared on fine grained attributes other than just equality
- Supports a rich set of data types
 rule Engines

• Reasoning Engines
  – Specify facts and rules than a set of instructions to run in a certain order like traditional languages
  – Based on logic programming technologies (Prolog, clips, lisp, ABLE)
  – Java compatible logic engines include (JRules, Quick Rules, Blaze Advisor)
  – Open source products (Jess, drools, info-sapient)
Rule Engines

- **ILOG Jrules**
  - Built on Java's versatility by providing extensive APIs and packaging them as comprehensive class libraries
  - Easy to integrate with Java application than ILOG JRules.

- **JESS**
  - Java expert system shell (Jess) is a programmer's library (written in Java)
  - Library serves as an interpreter for the Jess language

- **Infosapient**
  - Infosapient is a java-based product
  - Represents knowledge into computable terms using fuzzy logic

*(The best thing about this approach is its closeness to High-Level Policies of the IETF Policy specification approach)*
Implementation

• Chosen Verma’s Model
• Edited policies in XML
• Defined Schema
• Got License for Jess
• Open source/free
• Implementation Problems
  – Conversion of XML to Jess language
Questions