User-Centric QoS Policies: or Saying what And how

Mark Bearden & Sachin Garg
Avaya Labs Research, Avaya Inc.

Woei-Jyh Lee
University of Maryland

Aad van Moorsel
HP Labs, Hewlett Packard
The Problem

1. Enterprises need **service-level QoS management**:
   - Ensure “critical” services (e.g. DNS, billing) are available to critical clients, despite changing state of the network and resources
     - e.g., Clients of Directory or ERP servers need “fast” response time: $90\% < 150\text{ms RTT}$ (response time)
   - Charge premium for guarantee of performance/availability
     - e.g., Client pays extra $$ for providing <5\% request loss rate
   - Reduce “over-engineering” of the network

2. Current Policy based solution:
   - Administrator performs “ad hoc magic” to translate user level goals (<5\% loss rate) to static rules (give user1 higher priority than user2)

3. Limitations of current solutions:
   - Does not lend itself to verification
   - *Static* configuration (provisioning) of network elements
   - *Passive* management; reliance on manual decisions when alarms generated; administrator must manually write recovery scripts, reprovision, etc.
Policy-Based Management (PBM)

1. "How": IETF defines management **policy** (desired behavior) as if condition then **action rules**
   - if ( (usr = Fred) & (app = Web)) then ( Priority = 5 )
   - Impractical for user (Fred) to specify rules

2. "What": Support intuitive **propositional** Layer-7 QoS specification ("goals")
   - Fred wants response time of less than 3s (90% of the time) for Web (or XMS, SAP, ...)
   - Fred tells system administrator; administrator inputs **goal** to management server:
     Satisfy 90%respTime<3 for user Fred and service Web
   - Server derives **policy rules** for devices:
     if QueuedPacket(Fred's IPaddr, Web) then set Diffserv CP=3
     if QueuedPacket(Fred's IPaddr, Web) then move to Priority 2 Queue
Our Terminology

1. **Policy** -
   - Informal: Management strategy or procedure
   - Formal: A parameterized management function
     - Inputs: Monitored state
     - Outputs: Control signals, alarms
     - Parameters: *Domain* and *Objective*

2. **Domain** - Set of services, clients, and/or resources (policy targets)
   Ex.: SAP server, SAP clients, intra-network elements, hosts in server farm

3. **Objective** - Expression of desired client/service interaction (usually Quantitative)
   Ex.: < 10% packet loss for SAP clients
**Policy Program** - enforces a policy function

**Policy Component**
- Re-usable software module ("policy bean")
- Examples: Monitor, Evaluator, Action components

**Policy Package**
- Set of components (Java objects) that enforce one policy
Policy Package

1. **Goal Template:** “Enforce \( G \) for user \( U \) and service \( S \) in domain \( D \).”
   - \( G = \{ \{ \text{AvgResponseTime} \}, \{<,>\}, \{\text{Integer (seconds)}\} \} \)
   - \( U = \{\text{IPhost}\} \)
   - \( S = \{\text{SAP}, \text{HTTP}\} \)

2. **Policy Logic - \( f(U,S,T,D) \):**

   ```
   if (monitor_time_elapsed \geq \text{monitor interval})
   \text{AvgResponseTime}(U)
   
   \text{if ( AvgResponseTime}(U) < T )
   \text{compute } H_{\text{best}} = \text{“best” server } S \text{ for } U
   \text{compute } P_{\text{best}} = \text{priority for } U\text{’s requests}
   \text{remap } U \text{ requests to } H_{\text{best}}
   \text{set routers in } D: \text{ set traffic}(U) = P_{\text{best}}
   ```
Example Goal Specification

Goals are propositions with form:

Satisfy \texttt{GOAL\_EXPRESSION} for user U and service S for domain D

\textbf{GOAL EXPRESSION :}

- \textit{Metric: ResponseTime}
- \textit{Operator: “<”}
- \textit{Value syntax: 3ms}

- U: \textit{user type} (e.g. IPhost)
- S: \textit{service type} (e.g. HTTP server)
- D: \textit{enforcement domain elements} (e.g. Diffserv-aware routers, Layer-7 switches, etc.)
Policy Program Definition

1. **Goal Template:** “Enforce $G$ for user $U$ and service $S$ for domain $D$.”

2. **Example of Goal “Types”:**
   - Satisfy “AvgResponseTime < 1s” for user 135.104.25.50 and service HTTP in network domain *birdnet*
   - Satisfy “AvgResponseTime <= .5s” for user 135.104.25.50 and service SAP in network domain *birdnet*
   - Network domain *birdnet* contains Diffserv-aware routers

3. **Constraints:**
   - $G = \{ \{\text{AvgResponseTime}\}, \{<,\leq,\geq,>\}, \{\text{Integer (seconds)}\} \}$
   - $U = \{\text{IPhost}\}$
   - $S = \{\text{SAP, HTTP}\}$
   - $D = \{\text{Diffserv-aware routers}\}$
Management Server

Console (GUI)

Component Loader

Load/Unload

Initialize

Policy Programs

Domain and Policy data

Read/Write State Callbacks

(A)periodic Tasks

Scheduler

Policy Program Repository

(System Model
(Active)

(Java VM)
Core Service: System Model

• Object-oriented database
  • Targets and Domain Objects (Domain = Groups of Targets)
  • Policy Goal Objects
  • Policy Rule Objects (IETF-compliant)
  • Constraint Objects
    • *Types and allowed associations* for Domains, Targets, and Goals (meta-data)
Core Service: System Model

- Object-oriented database
- Targets and Domain Objects (Domain = Groups of Targets)
- Policy Goal Objects
- Policy Rule Objects (IETF-compliant)
- Constraint Objects
Customizable PBM Architecture

1. Rapid Development via Reuse and Customization
   - Policy server (reusable) and “plug-in” packages (domain-specific)
   - Packages contain reusable components (Java bean-like objects) and domain-specific logic
   - Examples: Monitoring/control of different devices, vendors, protocols; with different correlation and device control logic

2. Key Reusable Components:
   - “Skeleton” (or “horizontal framework”) for family of QoS management servers:
     $ $ OO System Model
     $ $ Graphical Interface for Defining Domains and Goals
     $ $ Package/Component Loader for Run-time or Boot-up Customization

3. Custom Logic Can be Reused via Components
   - Component and server API’s
   - (Ongoing) Library of developed components
Existing Monitoring Products

1 Application Level QoS monitoring
   - Lucent (VitalSuite)
   - Ganymede (Pegasus)
   - Concord (eHealth)
   - Mercury Interactive
   - Micromuse (NetCool)
   etc.

1 Network State monitoring
   - Lucent (VitalNet)
   - Hewlett Packard (Openview)
   - Nortel (Optivity)
   - Cisco
   etc.
Existing Control Products
(Policy-Based Management)

1 Vendors:
   - Avaya (Cajun Rules), Cisco (QPM), Nortel (Optivity Policy Services), IPHighway (thruQoS), Orchestream, etc.

1 “Policy”: if condition then action rules
   - if (usu = Fred) & (app = Web) then (Priority = 5)

1 Value and Differentiation:
   - Device support
   - Protocol support (COPS, SNMP, CLI, etc.)
   - Set of conditions and actions
   - Directory support
   - Architecture (distributed versus centralized)

1 Current Limitations:
   - 1. Rules specify low-level control actions on network devices
   - 2. No support for application-level QoS specification and enforcement
   - 3. No feedback of network state for dynamic policies
“QoS Server” Prototype Architecture

Gallifrey Management Server + VS, RNR Plug-ins (Java)

VitalSuite Status Repository (SQL)

Policy Rule repository (LDAP)

Service/Network QoS Measurements

Policy targets

Policy Decision Point (RNR Server)

CLI, Telnet, SNMP

QoS Goal Specification Console

RNR Administrator Console

VitalSuite Server

Active database

Policy targets

DSOM 2000, Austin, Texas
Page 16
12/5/2000
Potential “QoS Server” Features

1 Manual Management
   - Report to Sys-Admin which Policy Rules responsible for “QoS misses”
     $ Identify in GUI
       $ “missed” QoS goals
       $ rules that map to missed goals
     $ Tabular reports identifying
       $ rules mapped to QoS misses
       $ rules mapped to under-and over-provided QoS

1 Automated QoS/SLA Management
   - Auto-generate modifications to Policy Rules (write/modify rules in the LDAP directory)
     $ user-approved
     $ user-overridable
     $ automatically applied
   - Integration with Capacity Planning and Billing Software
Goals and Rules Schema (Partial)
QoS Server User Interface
Example: Report Unsatisfied Goals

“Real-time” indication:
• White: Satisfied
• Red: Not Satisfied