TECHNISCHE UNIVERSITÄT
CAROLO-WILHELMINA
ZU BRAUNSCHWEIG

Network Monitoring with Asynchronous Notifications in Web Service Environments

Torsten Klie
Technische Universität Braunschweig
Institut für Betriebssysteme und Rechnerverbund

Florian Müller
Universität Düsseldorf
Institut für Informatik

Stefan Fischer
Universität zu Lübeck
Institut für Telematik

Outline

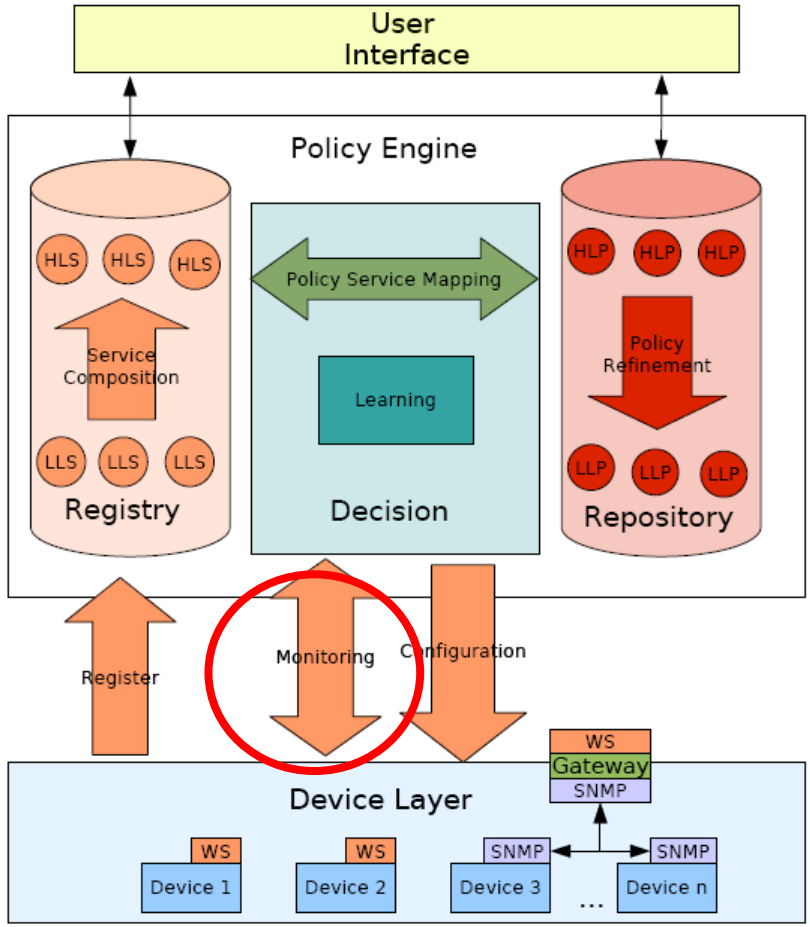
- **Introduction**
  - Motivation
- **Comparison of Existing Approaches**
  - Traditional Approaches
    - SNMP Traps
    - Syslog
  - Approaches based on Web-Services
    - WS-Eventing
    - Oasis WSDM
    - Netconf
- **Our Approach**
  - Design Overview
  - Implementation Details
- **Conclusions and Outlook**

Motivation

- **Growth of the Internet**
  - Dependency on a reliable communications infrastructure is growing
  - Complexity of networks is growing
- **Monitoring large networks**
  - Polling
    - Wasting resources if state does not change often
    - Hearbeat effect
  - Asynchronous notifications
    - Inform managers about important events
      - Failures
      - State changes
    - Management by exception
- **Our Requirements**
  - Independent from management frameworks
  - Support for Web services

Web Services for Network Management

- **Basis for future management infrastructures**
- **Common Standards**
- **SOA effectively used in e-business domain**
  - Divisions
  - Company aquisitions
  - Management processes are not different than business processes
- **Web services composition**
  - Can be automated with Semantic Web Technology
  - Complementary technique to policy refinement that can help to make management more autonomic



Comparison of Existing Approaches

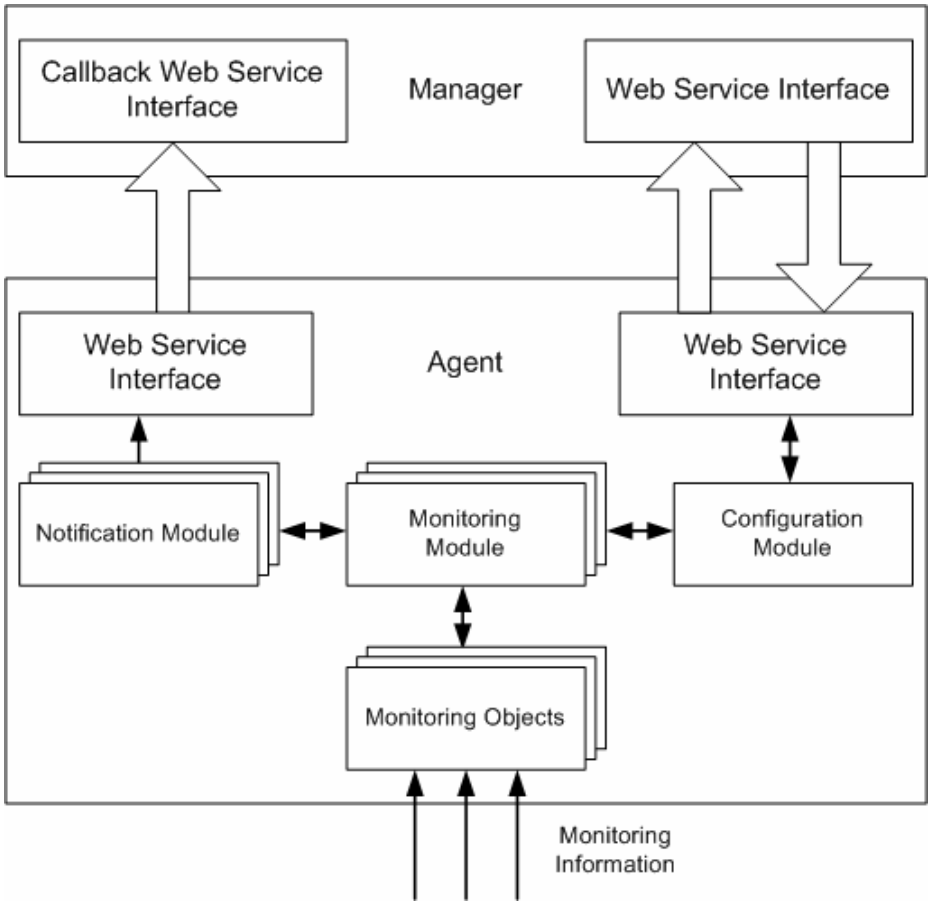
	SNMP Traps	Syslog	WS-Eventing	WS-Base Notification	NETCONF	Our App.
Scope	network devices	system applications	generic	generic	network devices	generic
Framework	SNMP	none	WS-*	WSDM	NETCONF	none
Data Format	ASN.1 (BER)	syslog	XML	XML	XML	XML
Transport Protocols	UDP	UDP, others	SOAP (HTTP, HTTPS)	SOAP	SSH, BEEP, SOAP	SOAP
Web Services Support	no	no	yes	yes	optional	yes
Intermediaries	no	yes	no	yes	no	no
Subscription	no	no	yes	yes	yes	yes
Automatic Subscription Termination	n/a	n/a	timeouts	no	no	no
Transport Modes	push	push	push, others	push, pull	push	push
Event Classes	7	24	user	user	user	5+user

Our Design

- **Manager-agent paradigm**
  - Manager has GUID
  - Agent integrated or proxy
- **Subscriptions**
- **Monitoring functions**
  - Statistical functions
  - Event functions
- **Example:**
  - Object: „processor load“
  - Condition: „> 3.0“
- **Parameters depend on use case and type of monitoring function**

- **Event tags:**
  - `fault`
  - `information`
  - `state change`
  - `configuration`
  - `periodic notification`

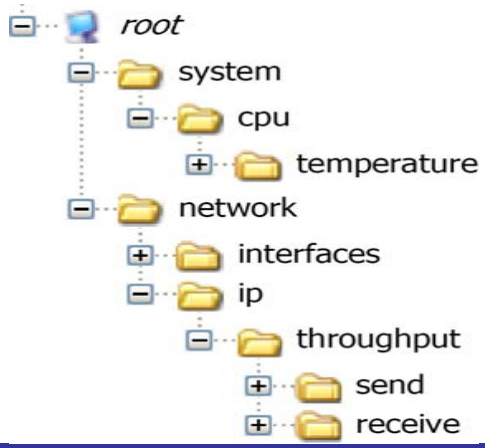
Design



- **Configuration interface**
  - `subscribe (guid, mf, params)`
  - `unsubscribe (guid, handle)`
  - `reconfigure (guid, handle, params)`
  - `getMonitoringObjects()`
  - `getParameterInformation (mf)`
  - `getRegisteredMonitoringInstances (guid)`
- **Managers must provide call-back interface (`pushNotification (guid, handle, msgClass, data)`)**

Implementation

- **Manager and agent prototype in Java 5.0**
- **Monitoring functions**
  - Implemented in `agent.monitorObjects`
  - Arbitrary nesting levels (e.g. `agent.monitorObjects.system.cpu.Temperature`)
  - Reflection



- **Monitoring function implements monitor_run()**
- **Generic monitoring thread in each agent**
  - Calls monitor_run() periodically
  - Can be overridden, if needed
  - One thread / manager (+ corresponding notifications thread)
- **Manager Application**
  - GUI
  - Control subscriptions
  - Display notifications

Evaluation

- **Debugging & JUnit tests**
- **Experiments with several scenarios**
  1. 1 manager ↔ 1 agent
     - Tests of configuration functions
     - Sending and receiving notifications
  2. n managers ↔ 1 agent: Checks for data isolation
  3. 1 manager ↔ n agents: Generalization of (1)
  4. n managers ↔ m agents: Most realistic tests
- **Initial tests: good performance**
- **More realistic tests (and comparisons with other approaches)**

Summary and Outlook

- **Summary**
  - Compared traditional and new notification approaches
  - Discussed some details of our own notifications approach
  - Advantages
    - Management by delegation
    - Management by exception
    - Not bound to any management framework
  - Disadvantages
    - Larger footprint on agents
    - Larger foorprint due to managent by delegation
    - Not bound to any management framework
- **Future Work**
  - Analyze interoperability problems
  - Investigate integrative solutions
  - Integrate prototype in our autonomic management architecture

The End

- **Questions or comments?**
  - Here and now: speak up!
  - Via e-mail to `tklie@ibr.cs.tu-bs.de`