Push vs. Pull in Web-Based Network Management

IM'99, Boston, MA, USA May 24–28, 1999

Jean-Philippe Martin-Flatin
Swiss Federal Institute of Technology, Lausanne
Institute for computer Communications and Applications
ICA
martin-flatin@epfl.ch
http://icawww.epfl.ch/~jpmf/
Outline

• Problems with SNMP-based network management
• Proposed solution:
  ■ Web-based network management (HTTP, Java applets and servlets)
  ■ push model for regular management
  ■ pull model for ad hoc management
• Overview of JAMAP
• Conclusions
Today’s management of IP networks

- SNMP frameworks (v1, v2c, v3)
  - manager-agent paradigm
  - polling (pull model)
  - notifications (push model)

- SNMP protocols (v1, v2c, v3)

- Network Management Platforms (NMPs): HP OpenView, Cabletron Spectrum, IBM/Tivoli Netview, Sun Solstice...

<table>
<thead>
<tr>
<th>Mandatory tasks:</th>
<th>Optional tasks:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- network monitoring</td>
<td>- configuration mgmt</td>
</tr>
<tr>
<td>- data collection</td>
<td>- inventory mgmt</td>
</tr>
<tr>
<td>- notification handling</td>
<td>- ACLs mgmt</td>
</tr>
<tr>
<td></td>
<td>- billing...</td>
</tr>
</tbody>
</table>

- Vendor- or device-specific add-ons (e.g. CiscoWorks)
Problems with NMPs (1/2)

• For customers:
  ■ too expensive (hardware and software):
    ➤ dedicated hardware for network management
  ■ offer limited support for third-party RDBMSs
  ■ cost to migrate from Unix to Windows is too high:
    ➤ Unix expertise required to maintain existing platforms
    ➤ investment bound to processor & operating system

• For network equipment vendors:
  ■ the support of device-specific add-ons is too expensive:
    ➤ many NMPs
    ➤ many OSs
    ➤ many add-ons
Problems with NMPs (2/2)

• For customers and network equipment vendors:
  - poor time-to-market for add-ons:
    ➠ large vendors: several months after hardware release
    ➠ startups: never --> need separate NMPs (no integrated management)
  - MIB versioning:
    ➠ MIB upgrade in a network --> version mismatch between NMP and agents:
      - update NMP manually, device by device
        (no MIB-discovery protocol)
      - do not use new features of a MIB until all devices are upgraded
Problems with SNMP (1/2)

• SNMP expertise is scarce and expensive (esp. SNMPv3)
• Scalability, network overhead and latency are adversely affected by some early protocol design decisions (SNMPv1):
  ■ BER encoding
  ■ SNMP table retrieval mechanism (no get-table)
  ■ OIDs take much more space than values
  ■ no compression
• Low-level semantics:
  ■ aimed at instrumentation
  ■ no standard high-level APIs
  ■ site-specific network applications developed from scratch
  ■ bound to an NMP API, not a technology
Problems with SNMP (2/2)

• Security:
  ■ SNMPv1 and SNMPv2c: none; SNMPv3: not used yet
  ■ mgmt of remote subsidiaries (VPNs): expensive encryption hardware
  ■ firewalls: UDP relays

• Unreliable transport protocol:
  ■ important notifications (unacknowledged) are lost for silly reasons
  ■ SNMPv3 informs (acknowledged) are not used yet
  ■ important mgmt data requires retransmissions at the application level

• Distribution:
  ■ manager to manager: none (SNMPv2 M2M MIB obsolete)
  ■ manager to agent (mobile code): Script MIB not used yet

• Evolution hampered by legacy syst.: “better replace than repair”
Proposed Solution (1/2)

• Keep:
  ■ MIBs
  ■ organizational model

• Change management framework:
  ■ pull model --> push model for repetitive tasks
  ■ move some workload from the manager to the agents

• Change communication protocol:
  ■ SNMP --> HTTP
  ■ connectionless UDP --> persistent TCP connections
  ■ gzip compression
  ■ unlimited number of MIB variables per push cycle
  ■ BER encoding --> MIME parts + {strings, XML, ser. Java objects...}
  ■ natural table retrievals
Proposed Solution (2/2)

• Change NMP:
  ■ split manager:
    ➤ management server (Java servlets)
    ➤ management station (Web browser)
  ■ rewrite manager code: expensive binary software --> less expensive Java software (indep. of OS and proc., no RDBMS-specific glue code)
  ■ expensive specific add-ons --> less expensive standard Java applets
  ■ dedicated NMP hardware --> any hardware
  ■ few third-party RDBMSs --> any RDBMS via JDBC
  ■ distribution made easier:
    ➤ manager: monolithic NMP --> distributed servlets
    ➤ manager to manager: standard distributed Java application (future work)
    ➤ manager to agent (mobile code): object serialization (future work)
Why HTTP Between Agents and Managers?

• Four techniques to distribute a Java application:
  ■ HTTP
  ■ sockets
  ■ RMI
  ■ Java IDL (CORBA)

• Distributed objects in network management (RMI or CORBA):
  ■ telecoms = yes
  ■ Internet = no (maybe later: EmbeddedJava --> lightweight RMI)

• HTTP > sockets:
  ■ natural communication between servlets within the mgmt server
  ■ same technology within the server and between agents and server
  ■ firewall setup easier for nonexperts (e.g. Web server = mgmt server)
Why Use Push Technologies?

- Save bandwidth: decrease network overhead of mgmt data
- Example: error rate for inbound traffic through interface #3

- Move some load from the manager to the agents
- Pave the way to Management by Delegation:
  - delegate preprocessing to the agents
JAMAP: Monitoring and Data Collection

Management station

Management server

Agent

Event notification applet
Rule edition applet
MIB data subscription applet

Event manager servlet
Pushed data collector servlet
MIB data dispatcher servlet

push
push
push
push

J.P. Martin-Flatin
IM'99, May 24–28, 1999
JAMAP: Notifications

Management station

- Event notification applet
- Rule edition applet
- Notification subscription applet

Management server

- Event manager servlet
- Notification collector servlet

Agent

- Notification dispatcher servlet

Push arrows indicate the direction of notification flow.
Issues

• Firewalls: connection should be created by internal manager, not external agent

• Ensure persistent connections:
  ■ the agents must control the timeout value of their embedded HTTP server
  ■ the manager must reconnect in case of connection teardown

• Positions of client and server now reversed:
  ■ transfer of management data initiated by the agent
  ■ client side of the persistent connection still on the manager side
  ■ we want the server to initiate a transfer in a client-server architecture!
Positions of Client and Server Now Reversed

Manager

SNMP client

SNMP trap server

HTTP client

Agent

SNMP server

SNMP trap client

HTTP server

Polling

Notification

Inform

Push

SNMP

trap

client

161

162

162

80

162
HTTP and MIME

MIME = Multipurpose Internet Mail Extensions

- **Advantages:**
  - simple to implement
  - firewalls: minor change (assuming Web access already)

- **Drawbacks:**
  - the manager must detect a network outage to set up a new connection:
    - send keepalives if no data after 9 minutes
    - blind during 9 minutes, or send keepalives more often
Conclusions (1/2)

What do we gain by going from SNMP-based pull to Java-based push to manage IP networks?

- Get rid of the expensive NMP
- Use well-known Web technologies instead of domain-specific SNMP
- Reduce network overhead of management data
- Reduce development costs of add-ons
- Zero the time-to-market of add-ons (embedded)
- Put small and large equipment vendors in fair competition w.r.t. integrated management
- Simplify the management of remote subsidiaries across a firewall
- Improve the support for third-party RDBMSs
- Remain backward compatible by using proxies for legacy systems
Conclusions (2/2)

What does it cost to go from SNMP-based pull to Java-based push to manage IP networks?

- network equipment vendors must add software to their equipment:
  - HTTP server (usually done today)
  - push system
  - scheduling system
  - JDK (JVM)
- administrators need to synchronize the clocks of the managers and the agents (e.g. with NTP)
- we need professional-grade software for the manager:
  - more and more vendors in the Web-based management market