Web services in Twente

• Thomas Drevers: Performance, compared to SNMP
• Willem Wong: WS in mobile “phones”
• Hoda El Merabet: Application: “MRTG” in MS-Excel
• Pierre Humbert: Configuration: WS-Transactions
• Jeroen van Sloten: From SMI -> WSDL (MDA)
• + Others
Performance of web services compared to traditional SNMP

Thomas Drevers
Overview

• Introduction
  – Motivation
  – Performance
• Measurements
• Results
• Concluding remarks
• Questions
Motivation

– September 2002: IRTF-NMRG Osnabrück
– Web services versus SNMP
– Questions concerning web services performance
– We wanted to investigate this, and provide real figures
Performance

The performance is divided into three main areas

– Network usage
– System resource usage
  • CPU
  • Memory
– Total time of the operation
Overview

• Introduction
• **Measurements**
• Results
• Concluding remarks
• Questions
Measurements (1/5)

What is shown

– Figures that compare SNMP and web services implementation

– Two types of web services
  • Standard web services
  • Web services which use compression

– Two types of SNMP
  • Always: UDP
  • Sometimes: TCP
Measurements (2/5)

Implementation

– Net-SNMP
  limited MIBs

– gSOAP
  with Net-SNMP Data retrieval functions
Measurements (3/5)

Network usage

- For every operation two measurements are performed:
  1. Data in: IP + TCP/UDP + SOAP/SNMP layers
  2. Data in: the SOAP/SNMP layer
Measurements (4/5)

System resource usage

- Two kind of measurements:
  - CPU time consumption
  - Memory consumption
- CPU time consumption
  - Data retrieval
  - XML / BER processing (encoding / decoding)
- Memory consumption
  - program code
  - Permanent memory allocation (static)
  - Additional memory per operation (dynamic)
Measurements (5/5)

Total operation time

– The average amount of time the client has to wait until an operation is finished
# Measured Data: Model

## If Table

<table>
<thead>
<tr>
<th>ifIndex</th>
<th>ifDescr</th>
<th>ifMTU</th>
<th>ifSpeed</th>
<th>Etc …</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lo</td>
<td></td>
<td>100000000</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Eth0</td>
<td>1000000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Eth1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Measured Data: WSDL variants

- Cell
- Column
- Row
- Columns
- Table
Measured Data: WSDL Table (1)

<complexType name="ifEntry">
  <sequence>
    <element name="ifIndex" type="xsd:unsignedInt" minOccurs="1" maxOccurs="1"/>
    <element name="ifDescr" type="xsd:string" minOccurs="1" maxOccurs="1" nillable="true"/>
    <element name="ifType" type="xsd:unsignedInt" minOccurs="1" maxOccurs="1"/>
    <element name="ifMtu" type="xsd:unsignedInt" minOccurs="1" maxOccurs="1"/>
    <element name="ifSpeed" type="xsd:unsignedInt" minOccurs="1" maxOccurs="1"/>
    <element name="ifPhysAddress" type="xsd:string" minOccurs="1" maxOccurs="1" nillable="true"/>
    <element name="ifAdminStatus" type="xsd:unsignedInt" minOccurs="1" maxOccurs="1"/>
    <element name="ifOperStatus" type="xsd:unsignedInt" minOccurs="1" maxOccurs="1"/>
    <element name="ifLastChange" type="xsd:unsignedInt" minOccurs="1" maxOccurs="1"/>
    <element name="ifInOctets" type="xsd:unsignedInt" minOccurs="1" maxOccurs="1"/>
    <element name="ifInUcastPkts" type="xsd:unsignedInt" minOccurs="1" maxOccurs="1"/>
    <element name="ifInDiscards" type="xsd:unsignedInt" minOccurs="1" maxOccurs="1"/>
    <element name="ifInErrors" type="xsd:unsignedInt" minOccurs="1" maxOccurs="1"/>
    <element name="ifInUnknownProtos" type="xsd:unsignedInt" minOccurs="1" maxOccurs="1"/>
    <element name="ifOutOctets" type="xsd:unsignedInt" minOccurs="1" maxOccurs="1"/>
    <element name="ifOutUcastPkts" type="xsd:unsignedInt" minOccurs="1" maxOccurs="1"/>
    <element name="ifOutErrors" type="xsd:unsignedInt" minOccurs="1" maxOccurs="1"/>
    <element name="ifOutUnknownProtos" type="xsd:unsignedInt" minOccurs="1" maxOccurs="1"/>
  </sequence>
</complexType>
<complexType name="GetIfTableResponse">
    <sequence>
        <element name="ifEntry" type="utMon:ifEntry" minOccurs="1" maxOccurs="unbounded"/>
    </sequence>
</complexType>

<message name="GetIfTableRequest">
    <part name="community" type="xsd:string"/>
</message>

<message name="GetIfTableResponse">
    <part name="-sizeTable" type="xsd:int"/>
    <part name="ifEntry" type="utMon:ifEntry"/>
</message>

<portType name="GetIfTableServicePortType">
    <operation name="GetIfTable">
        <documentation>Service definition of function utMon__GetIfTable</documentation>
        <input message="tns:GetIfTableRequest"/>
        <output message="tns:GetIfTableResponse"/>
    </operation>
</portType>
Measured Data: C Structure Table

```c
struct ifEntry
{
    xsd__unsignedInt  ifIndex;   // interface index number
    xsd__string       ifDescr;   // interface description
    xsd__unsignedInt  ifType;    // Interface type
    xsd__unsignedInt  ifMtu;     // maximum packet size
    xsd__unsignedInt  ifSpeed;   // maximum speed
    xsd__string       ifPhysAddress; // Physical address of the interface (eg MAC)
    xsd__unsignedInt  ifAdminStatus; // preferred status set by admin (1=up, 2=down, 3=testing)
    xsd__unsignedInt  ifOperStatus; // operational status of the interface
    xsd__unsignedInt  ifLastChange; // Sysuptime of last change in operational status
    xsd__unsignedInt  ifInOctets;  // octets received by the interface
    xsd__unsignedInt  ifInUcastPkts; // unicast packets received
    xsd__unsignedInt  ifInDiscards; // Discarded incoming packets
    xsd__unsignedInt  ifInErrors;   // Erronous incoming packets
    xsd__unsignedInt  ifInUnknownProtos; // Amount of packets dicarded because of unknown protocols
    xsd__unsignedInt  ifOutOctets;  // number of outbound octets
    xsd__unsignedInt  ifOutUcastPkts; // number of outbound unicast packets
    xsd__unsignedInt  ifOutErrors;  // number of packets/units which could not be transmitted
};
```
Measured Data: SNMP variants

For SNMP the same variants as with WSDL are used

- Cell retrieves a single cell
  Uses the snmpget method

- Column retrieves a single column
  Uses the snmpgetbulk method with a maximum for the amount of rows

- Row retrieves a single row
  Uses a single snmpget method, with a request for each cell in the row

- Columns retrieves specified columns
  Uses the snmpgetbulk method with a request for each column and a maximum corresponding to the amount of rows

- Table retrieves complete table
  Uses a series of the snmpgetbulk methods to retrieve the whole table
Overview

• Introduction
• Measurements
• Results
• Concluding remarks
• Questions
Network usage (1/4)

IP and higher layer protocols
Network usage (2/4)

SOAP and SNMP protocols
Network usage (3/4)

Effect of more data (extra interfaces)
Network usage (4/4)

Conclusion

– For small amounts of data, SNMP is more efficient
– For large amounts of data, compressed web services are more efficient
– Uncompressed web services are not very efficient
CPU usage

The graph shows the time taken per operation for different types of services: normal web service, compressed web service, and SNMP. The operations are categorized as Cell, Column, Row, Columns, and Table. The time taken is measured in milliseconds (ms) on the y-axis, and the operations are shown on the x-axis.
CPU usage: Encoding and Decoding

![Bar chart showing time per operation for different operations and web services.](chart.png)
CPU usage: Data Retrieval

![CPU usage chart showing time per operation for different data retrieval operations: Cell, Column, Row, Columns, Table. The chart compares time per operation for web service and SNMP.]
CPU usage: Conclusions

- CPU usage primarily depends on data retrieval
- Our Web services implementation far better than (Net-)SNMP
- CPU usage of (Net-)SNMP grows linearly with the number of cells
- CPU usage of web services does not depend on the number of cells
- Compression takes roughly twice the time of normal encoding / decoding
Memory usage: static

• Program footprint
  – Approximately 1972 Kbyte for SNMP
  – Approximately 580 Kbyte for web services

• Amount of memory always allocated for data
  – SNMP 128 Kbyte
  – Web services 470 bytes
Memory usage: dynamic

![Graph showing memory usage for different operations.](image)

- **Operation**: Cell, Column, Row, Columns, Table
- **Memory Usage**: Dynamic
- **Memory Measurement**: Bytes/operation
- **Comparisons**: Normal web service, Compressed web service, SNMP
Memory usage: conclusions

Conclusion

– Net-SNMP uses more memory than our gSOAP web service implementation

– However, Net-snmp offers more functionality: comparison is therefore unfair
Total operation time

![Bar chart showing operation times for different types of operations and service types.]
Total operation time: conclusions

- Web services is much faster
- SNMP spends most of its time on (inefficient) data retrieval
- Message processing is not really the bottleneck
Overview

• Introduction
  – Web services and monitoring
  – Performance
• Measurements
• Results
• Concluding remarks
• Questions
Concluding remarks

– In our case study, performance of web services was better than SNMP, particularly when large amounts of data were retrieved

– The results will likely depend on the specific implementations

– Researchers who use performance as the reason to reject web services technology, make a mistake
BACKGROUND INFO: Message sequence

Web service message sequence

SNMP over UDP message sequence

SNMP over TCP message sequence