NETCONF, YIN and YANG, BEEP, and UML

Raj Jain
Washington University in Saint Louis
Saint Louis, MO 63130
Jain@cse.wustl.edu

These slides and audio/video recordings of this class lecture are at:
http://www.cse.wustl.edu/~jain/cse570-15/
Overview

1. NETCONF: Network configuration protocol
2. YANG and YIN: Data modeling
3. BEEP: Transport
4. UML: Software modeling
NETCONF

- IETF XML based Network device configuration protocol (RFC 6241, June 2011)
- Allows setting configuration parameters when the device is instantiated and changing these parameters later E.g., set IP address to 192.168.0.1
- Replacement for:
  - SNMP (Simple Network Management Protocol)
  - Command line interfaces (CLIs)
  - Scripts used by operators
- XML based ⇒ Both human and machine readable
- Also allows monitoring the device
- Uses remote procedure calls (RPCs) called “Operations”
- Runs over SSH ⇒ Secure

Ref: https://en.wikipedia.org/wiki/NETCONF#Operations
Washington University in St. Louis http://www.cse.wustl.edu/~jain/cse570-15/
©2015 Raj Jain
NETCONF Protocol Layers

- **Content**: Full configuration, Partial Configuration
- **Notification Data**: Edit, Copy, Delete
- **Operations**: RPC, RPC-Reply, Notifications (Publish/Subscribe)
- **Messages**: SSH, TLS, BEEP over TLS, SOAP over HTTP over TLS

- **Notification**: Publish/subscribe mechanism to get state/alerts

Configurations

1. **Running**: Complete currently running configuration
2. **Start up**: Configuration to be used on next reboot
3. **Candidate**: Part of currently running configuration. Scratch pad for configuring pieces before commit.

```
Candidate

--------------------
|                  |
| Commit           |
| Running          |
| Copy             |
| Start up         |
```

©2015 Raj Jain
NETCONF Operations

- Get: Get complete running configuration and state
- Get-Config: Get all or part of running configuration
- Edit-Config: Edit configuration
- Copy-Config: Copy the entire configuration store to another
- Delete-Config:
- Lock: Lock the full/partial configuration
  (so that no one else can modify)
- Unlock
- Close-Session: Graceful termination of session
- Kill-Session: Abort
NETCONF Parameters

- Parameters are stored in a hierarchical XML file.
- Any branch or the entire tree can be over-written or retrieved.

Washington University in St. Louis  
http://www.cse.wustl.edu/~jain/cse570-15/  
©2015 Raj Jain
Show currently running BGP Configuration:

```xml
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get-config>
        <source>
            <running/>
        </source>
        <filter type="subtree">
            <top
                xmlns="http://example.com/schema/1.2/config">
                <bgp/>
            </top>
        </filter>
    </get-config>
</rpc>
```

Remote Procedure Call
Name Space
Operation
Which configuration?
Running configuration
What subpart of configuration?
User defined name space schema
BGP subpart

Washington University in St. Louis http://www.cse.wustl.edu/~jain/cse570-15/ ©2015 Raj Jain
YANG Data Model

- Yet Another Next Generation data modeling language. By IETF netmod working group
- Sequel to SMI (Structure of Management Information) used with SNMP, SMIv2 used by SNMPv2, and SMIng
- To express configuration data and state data
- **Data model**: Describes the data, its *constraints*. A.k.a., *Schema*
  - E.g., address may consist of street, state, zip *within* 50 states. 1 Brookings Dr., Saint Louis, MO 63130 is an *instance*.
- YANG defines a number of built-in data types and specifies a way to construct more complex data types.

YANG is used for configuration data, state data, RPCs (Operations), and event notifications.
YANG Node Types

- **Container**: A subtree of related nodes. No data values. Only a set of child nodes. Single instance. E.g., WUSTL

- **Leaf**: Has a value and no child nodes. Can have a default value. Can be mandatory or optional. Single instance. E.g., Department of CSE

- **List**: A set of list entries. Each list entry may contain many child nodes including other lists. Uniquely identified by its key value. E.g.,
  
  ```
  list user {
    key login-name;
    leaf login-name { type string; }
    leaf full-name { type string; }
  }
  ```
YANG Node Types (Cont)

- **Leaf-List**: A set of leaves. Used to define a sequence. Leaf nodes have only one instance, while leaf-lists may have multiple instances, e.g., “Department” is a leaf-list, while “CSE” is a leaf in “WashU” container.

- **Typedef**: Define new types by adding to another data type
  ```
  typedef port-number {
      type uint16 {
        range “1..65535”;  
      }  
  }  
  ```

- **Uses**: Refines and augments another data type
  ```
  Container server {
      Container address {
        uses address-type;  
      }  
  }  
  ```
YANG Node Types (Cont)

- **Anyxml**: any chunk of XML data
- **Choice**: One of \( n \) case statements. Only one is satisfied
- **Augment**: Allows vendors to add vendor-specific data to standard data modules. Should not break applications that do not understand vendor-specific data

![Diagram of YANG Node Types]

Ref: M. Bjorklund, “YANG HighLevel Presentation, YIN, XML on the wire,” IETF 72,
# Built-in Data Types in YANG

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>binary</td>
<td>Any binary data</td>
</tr>
<tr>
<td>bits</td>
<td>A set of bits or flags</td>
</tr>
<tr>
<td>boolean</td>
<td>“true” or “false”</td>
</tr>
<tr>
<td>int(n)</td>
<td>n-bit signed integer, (n=8, 16, 32, 64)</td>
</tr>
<tr>
<td>uint(n)</td>
<td>n-bit unsigned integer, (n=8, 16, 32, 64)</td>
</tr>
<tr>
<td>decimal64</td>
<td>64-bit signed decimal number</td>
</tr>
<tr>
<td>string</td>
<td>Human readable string</td>
</tr>
<tr>
<td>empty</td>
<td>A leaf that does not have any value</td>
</tr>
<tr>
<td>enumeration</td>
<td>Enumerated strings</td>
</tr>
<tr>
<td>intentityref</td>
<td>A reference to an abstract identity</td>
</tr>
<tr>
<td>instance-identifier</td>
<td>References a data tree node</td>
</tr>
<tr>
<td>leafref</td>
<td>A reference to a leaf instance</td>
</tr>
<tr>
<td>union</td>
<td>Choice of member types</td>
</tr>
</tbody>
</table>
### YANG Examples

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>leaf host-name{</code>&lt;br&gt;<code>  type string; }</code></td>
<td>Built-in data type</td>
</tr>
<tr>
<td><code>typedef percent {</code>&lt;br&gt;<code>  type uint8 {</code>&lt;br&gt;<code>  </code>  range “0..100” ;}`</td>
<td>User defined data type</td>
</tr>
<tr>
<td><code>leaf-list domain-search{</code>&lt;br&gt;<code>  type string;}</code></td>
<td>List of domains to search</td>
</tr>
<tr>
<td><code>container food {</code>&lt;br&gt;<code>  choice snack {</code>&lt;br&gt;<code>  </code>    case sports-arena {<code>&lt;br&gt;</code>      leaf pretzel {<code>&lt;br&gt;</code>  <code>        type empty; }</code>&lt;br&gt;<code>    case late-night {</code>&lt;br&gt;<code>  </code>      leaf chocolate {<code>&lt;br&gt;</code>  <code>        type enumeration {</code>&lt;br&gt;<code>          enum dark;</code>&lt;br&gt;<code>          enum milk;</code>&lt;br&gt;<code>          enum first-available;</code>&lt;br&gt;<code>        }</code>&lt;br&gt;<code>    }</code>&lt;br&gt;<code>  }</code>&lt;br&gt;<code>  }</code>&lt;br&gt;<code>  }</code>&lt;br&gt;<code>  }</code>&lt;br&gt;<code>  }</code>&lt;br&gt;<code>  }</code>&lt;br&gt;<code>  }</code>&lt;br&gt;<code>  }</code>&lt;br&gt;<code>  }</code>&lt;br&gt;`</td>
<td>No value. Only presence or absence.</td>
</tr>
<tr>
<td></td>
<td>Enumeration = Set of assigned names</td>
</tr>
<tr>
<td></td>
<td>Member of enumeration</td>
</tr>
</tbody>
</table>
YIN

- Yin and Yang: Complementary and interrelated
- YANG is human readable
  YIN is an equivalent XML syntax for YANG.
- YANG module can be translated into YIN, manipulated by XML tools and translated back into YANG without losing any information.

  \[ \text{YANG}(\text{YIN}(M)) == M \]

- XML syntax useful for XML tools.
  E.g., Extensible Stylesheet Language Transformations (XSLT)
  - Extract documentation
  - Generate Code
  - Display graphically
YIN Example

- **YANG:**
  
  `leaf address {
      type inet: ip-address;
  }
  
- **YIN:**
  
  `<leaf name="address">
      <type name="inet:ip-address" />
  </leaf>`
YANG Tools

- **Libsmi**: Generate YANG from SMIv2
- **Pyang**: Validate YANG. Translate between YANG and YIN. Generate XML schema definition (XSD) and document scheme definition language (DSDL) from YANG or YIN.
- **Yangdump**: Validate YANG. Generate XSD and HTML from YANG.

Secure Transports

- SSH: Secure Shell
- TLS: Transport Level Security
- BEEP: Blocks Extensible Exchange Protocol
  - Framework for creating network application protocols
  - Provides building blocks, e.g., authentication, framing, pipelining, multiplexing, reporting, …
  - Allows multiple parallel pipelines (channels)
  - Can define multiple profiles (sets of blocks)
  - Runs on TLS
- SOAP: Simple Object Access Protocol
  - Protocol to exchange structured information for web services
  - Can run over HTTP, SMTP, TCP, UDP, …

BEEP

- Blocks Extensible Exchange Protocol
  A general purpose protocol framework for exchange of data.
- Allows application developers to concentrate on their application messages and offload message exchange by using ready made BEEP code.
- BEEP implementations in C, Java, Pascal, C++, Python, JavaScript, Ruby, TCL, perl are available at beepcore.org
- Like XML, BEEP is eXtensible and most applications can be implemented on the top of BEEP.
- After the success of HTTP, many applications started modifying HTTP to suite their applications, e.g., Internet Printing Protocol added a few new headers to HTTP.
- HTTP is a stateless client-server protocol. Not easy to use for stateful or peer-to-peer applications.

Washington University in St. Louis http://www.cse.wustl.edu/~jain/cse570-15/ ©2015 Raj Jain
BEEP (Cont)

- BEEP is designed for applications that are:
  - Connection Oriented: Connect, Exchange, ..., Exchange, Disconnect.
  - Message Oriented: Loosely coupled peers communicating using messages
  - Asynchronous: Multiple parallel exchanges
    Example: HTTP, FTP, SMTP. Not good for one-shot exchanges, e.g., DNS

- BEEP Provides the following functions:
  - Separating one message from next (Framing)
  - Multiple parallel asynchronous exchanges
  - Negotiating encryption, authentication
  - Reporting errors

Ref: M. Rose, “The Blocks Extensible Exchange Protocol Core,” RFC 3080, March 2001,
http://www.ietf.org/rfc/rfc3080.txt?number=3080
BEEP

- *Initiator* sends a connection request to *listener* and sets up a bidirectional *session*. Multiple *channels* are then setup.
- *Channel 0* is used for managing other channels.
- Some *tuning* channels are used for negotiating profiles such as encryption and authentication for other *data exchange* channels.
- Libraries of standard profiles are available as XML Document Type Definitions (DTDs).
- Many application profiles use XML to encode their messages.

```
Channel 0 (Management)  
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>
Channel 1 (Profile A)  
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>
...                        
|               |               |
Channel n (Profile P, Q)  
```

Washington University in St. Louis http://www.cse.wustl.edu/~jain/cse570-15/ ©2015 Raj Jain
Unified Modeling Language (UML)

- UML is a modeling language for software engg
  Standardized by the Object Management Group (OMG) and ISO
- Structural diagrams show the static view of objects, attributes, operations, and relationships
- Behavior diagrams show the dynamic behavior in terms of collaborations among objects and state changes
UML Diagrams

1. Class Diagram: Attributes and relationships of systems classes
2. Component Diagram: System components and their dependencies
3. Composite Structure Diagram: Internal structure of a class
4. Deployment Diagram: Hardware used in implementations
5. Object Diagram: Structure of a sample modeled system
6. Package Diagram: Logical groupings inside a system
7. Profile Diagram: Profiles of various classes
8. Activity Diagram: Workflows of components in a system
9. UML State Machine Diagram: State transition diagram
10. Use Case Diagram: Actors and their goals in some use cases
11. Communication Diagram: Communication between components
12. Interaction Overview Diagram: Interactions between communication diagrams
13. Sequence Diagram: Sequence of messages between objects
14. Timing Diagram: Shows timing constraints

Washington University in St. Louis  http://www.cse.wustl.edu/~jain/cse570-15/
UML Diagram Notation

- Unidirectional Association: $A \rightarrow B$
- Bidirectional Association: $A \leftrightarrow B$
- Dependency: Change to $A$ will cause change to $B$ $A \longrightarrow B$
- Aggregation: $A$ is a part of $B$ $A \rightarrow B$
- Composite Aggregation: $A$ (Child) and $B$ (Parent) are tightly coupled such that child can not exist without the parent. $A \rightarrow B$
- Generalization: $A$ is a subclass of $B$ $A \rightarrow B$

Washington University in St. Louis http://www.cse.wustl.edu/~jain/cse570-15/ ©2015 Raj Jain
Sample UML Class Diagram for OF-Config Data Model

1. “OF capable switch” consists of many logical switches
2. OF capable switch is configured by many “OF configuration points”
3. “OF logical switch” is controlled by many “OF controllers”
4. Logical switch uses many “OF resources”
5. Resource types are Ports, Queues, External certificates, own certificate, and flow tables


Washington University in St. Louis http://www.cse.wustl.edu/~jain/cse570-15/ ©2015 Raj Jain
Summary

1. NETCONF is the network device configuration protocol (next generation of SNMP)
2. YANG is the human-readable data modeling language (next generation of SMI)
3. YIN is the XML version of YANG modules
4. BEEP is the message exchange transport protocol
5. UML is the software modeling language
Reading List

- Netconf central, http://www.netconfcentral.org/
WikiPedia Links

# Acronyms

- **BEEP**  Blocks Extensible Exchange Protocol
- **BGP**  Border Gateway Protocol
- **CLI**  Command line interface
- **CSE**  Computer Science and Engineering
- **DNS**  Domain Name System
- **DSDL**  Document Scheme Definition Language
- **DTD**  Document Type Definition
- **FTP**  File Transfer Protocol
- **HTML**  Hyper Text Markup Language
- **IETF**  Internet Engineering Taskforce
- **IP**  Internet Protocol
- **ISO**  Internet Standards Organization
- **NETCONF**  Network configuration protocol
- **OF-Config**  OpenFlow Management and Configuration Protocol
- **OF**  OpenFlow
- **OMG**  Object Management Group
Acronyms (Cont)

- RPC  Remote Procedure Call
- SMI  Structure of Management Information
- SMIng  Structure of Management Information Next Generation
- SMIv2  Structure of Management Information Version 2
- SNMP  Simple Network Management Protocol
- SNMPv2  Simple Network Management Protocol Version 2
- SOAP  Simple Object Access Protocol
- SSH  Secure Shell
- TCL  Tool Command Language
- TCP  Transmission Control Protocol
- TLS  Transport Layer Security
- UDP  User Datagram Protocol
- UML  Unified Modeling Language
- WashU  Washington University in Saint Louis
- WUSTL  Washington University in Saint Louis
- XML  eXtensible Markup Language
Acronyms (Cont)

- XSD: XML Scheme Definition
- XSLT: Extensible Stylesheet Language Transformations
- YANG: Yet Another Next Generation Data Modeling Language
- YIN: Complement of Yang