tail-f

Network Automation

Carl Moberg, <calle@tail-f.com>
@cmoberg

Speaker Background

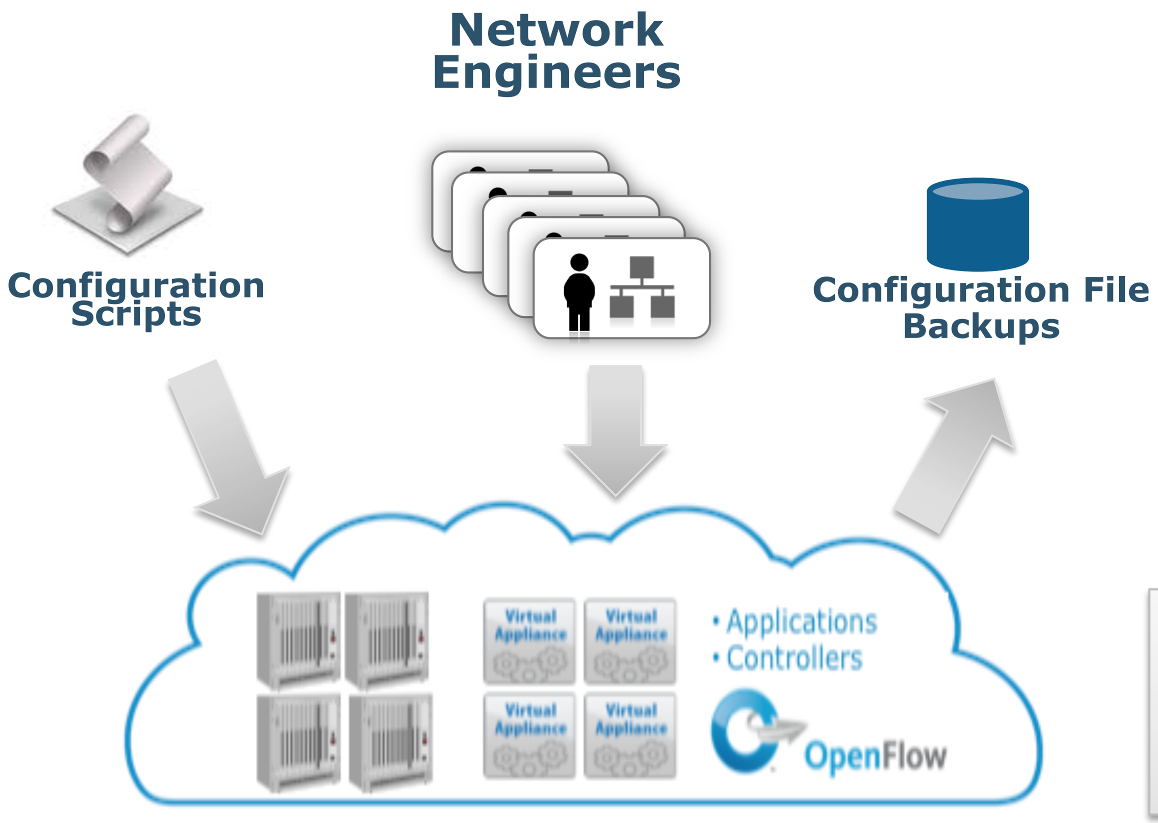
- Long time ago:
  - AS 3301, 1299
  - postmaster@telia.com
- Lately:
  - 2001-2006: tried to solve user and service management for public WiFi
  - 2006-2009: tried to solve on-device configuration management (ongoing)
  - 2009-present: working to solve network automation and programmability
- Moved from Stockholm to San Jose to mess better with people

Carl Moberg
@cmoberg

New year, new vanity plates. #NETCONF pic.twitter.com/eclanNjDxh

Today’s Topic: #1 Market Leader in Configuration Management

Network Configuration



Works fine **iff**:

- Frequency of change low
- Complexity of change low
- Cheap to fail

"It's not a bump or a hurdle, it's a brick wall"
-- *MSO on Ethernet Operations*

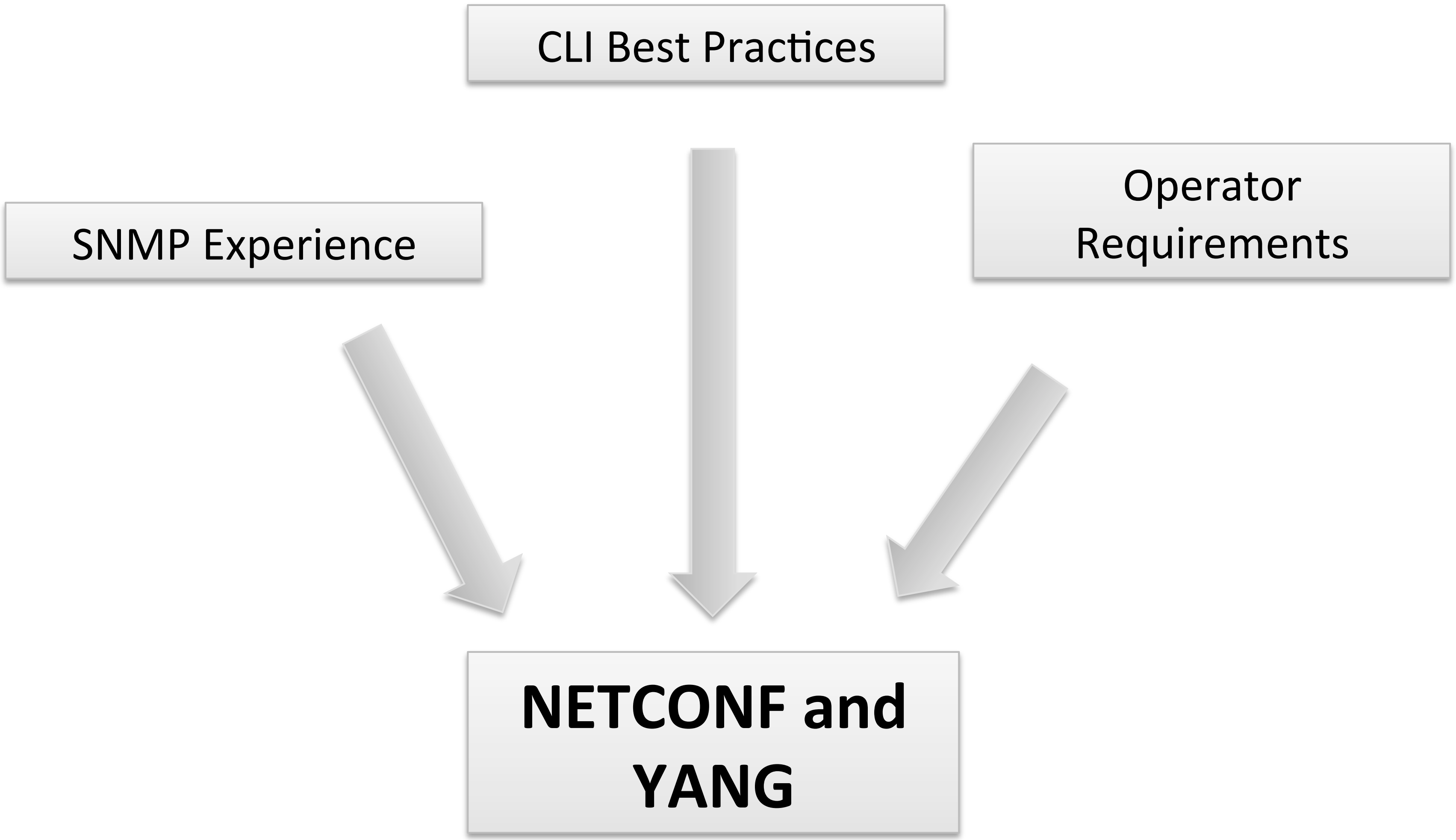
Origins of NETCONF and YANG (the Beginning)

- Several meetings at events in 2001 (NANOG-22, RIPE-40, LISA-XV, IETF 52)
  - Operators expressing opinion that the developments in IETF do not really address requirements configuration management.
- June of 2002, the Internet Architecture Board (IAB) held invitational workshop on Network Management [RFC3535] to
  - Identify a list of technologies relevant for network management with their strengths and weaknesses
  - Identify the most important operator needs.

Personal Favorites from RFC 3535 (> 10 years ago!)

- It is necessary to make a clear distinction between configuration data, and data that describes operational state and statistics.
- It is necessary to enable operators to concentrate on the configuration of the network as a whole rather than individual devices.
- Support for configuration transactions across a number of devices would significantly simplify network configuration management
- A mechanism to dump and restore configurations is a primitive operation needed by operators
- There is no common database schema for network configuration, although the models used by various operators are probably very similar. It is desirable to extract, document, and standardize the common parts of these network wide configuration database schemas.

Best Practices Coming Together



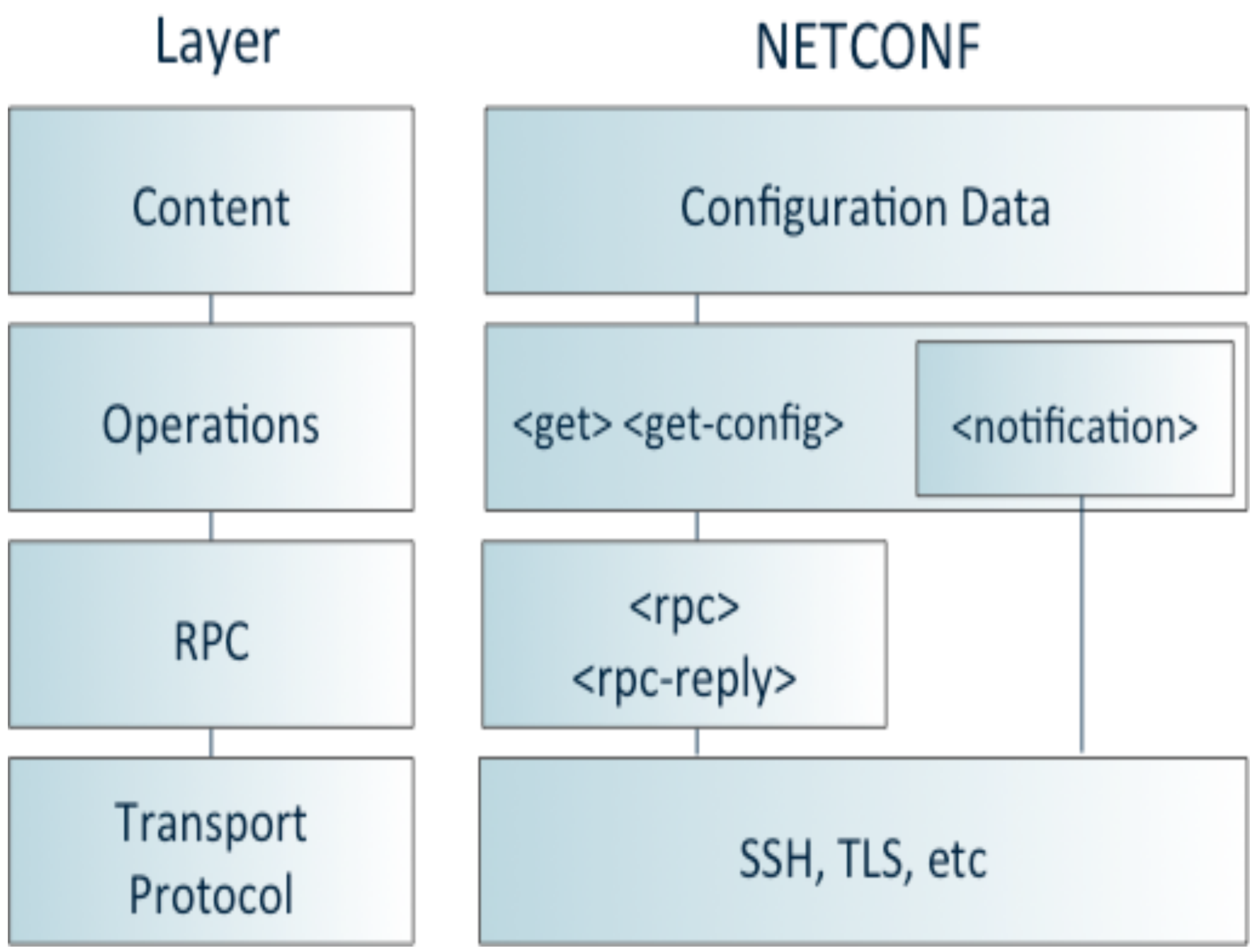
NETCONF – A Protocol to Manipulate Configuration

- IETF network management protocol
- Distinction between configuration and state data
- Multiple configuration data stores (candidate, running, startup)
- Configuration change validations
- Configuration change transactions
- Selective data retrieval with filtering
- Streaming and playback of event notifications
- Extensible remote procedure call mechanism

Why you should care:

NETCONF provides the fundamental programming features for comfortable and robust automation of network services

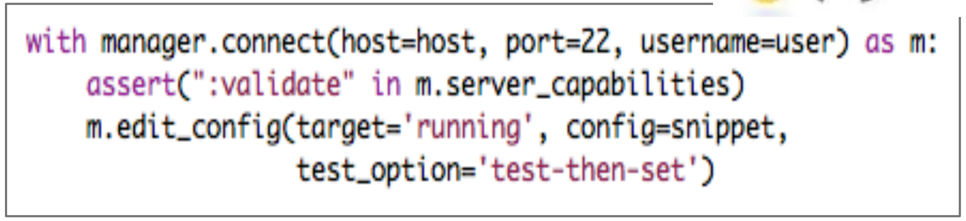
NETCONF Layering Model



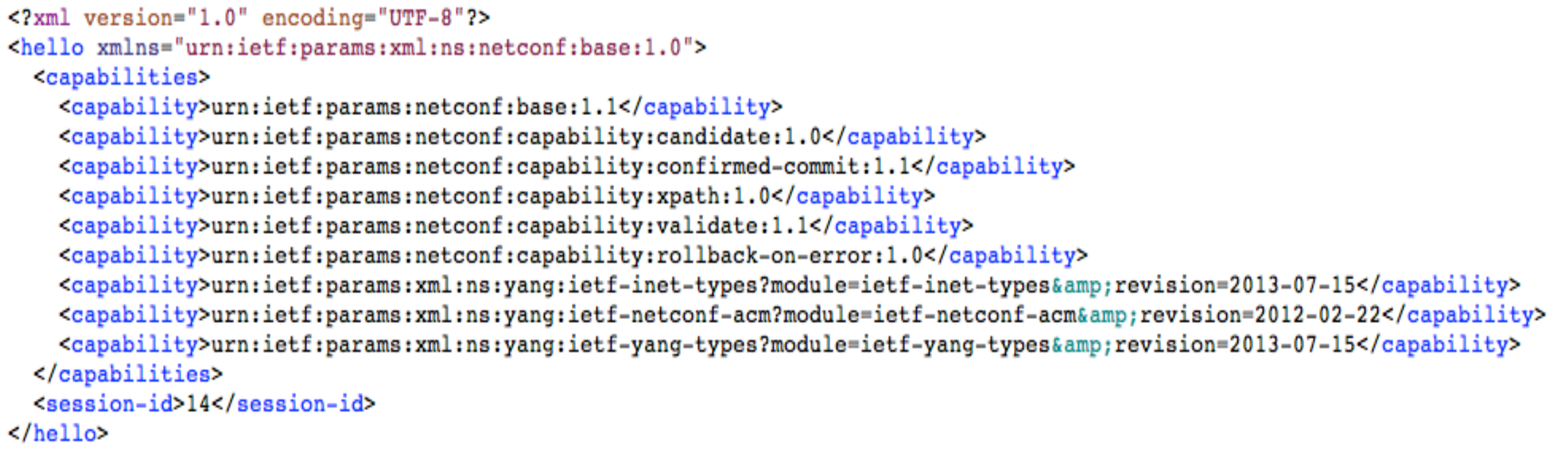
NETCONF Capabilities

- A capability is a set of functionality that supplements base NETCONF spec
- Capabilities augment:
  - Additional operations
  - Content allowed inside these operations
- Capabilities advertised by server during session establishment

- Base NETCONF specification provides very restricted set of operations for lightweight server implementations

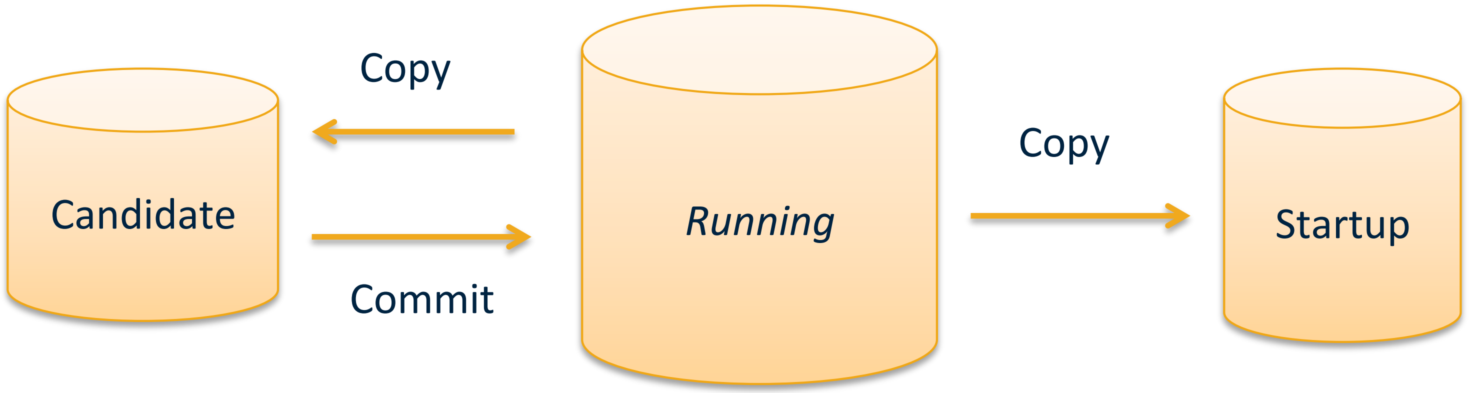


```python
with manager.connect(host=host, port=22, username=user) as m:
    assert(":validate" in m.server_capabilities)
    m.edit_config(target='running', config=snippet,
                  test_option='test-then-set')
```



```xml
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.1</capability>
    <capability>urn:ietf:params:netconf:capability:candidate:1.0</capability>
    <capability>urn:ietf:params:netconf:capability:confirmed-commit:1.1</capability>
    <capability>urn:ietf:params:netconf:capability:xpath:1.0</capability>
    <capability>urn:ietf:params:netconf:capability:validate:1.1</capability>
    <capability>urn:ietf:params:netconf:capability:rollback-on-error:1.0</capability>
    <capability>urn:ietf:params:xml:ns:yang:ietf-inet-types?module=ietf-inet-types&amp;revision=2013-07-15</capability>
    <capability>urn:ietf:params:xml:ns:yang:ietf-netconf-acm?module=ietf-netconf-acm&amp;revision=2012-02-22</capability>
    <capability>urn:ietf:params:xml:ns:yang:ietf-yang-types?module=ietf-yang-types&amp;revision=2013-07-15</capability>
  </capabilities>
  <session-id>14</session-id>
</hello>
```

NETCONF Conceptual Databases



The optional **Candidate Datastore** represents a working copy for manipulating configuration data with no impact on current configuration

The mandatory **Running Datastore** represents the complete and active configuration on the network device

The optional **Startup Datastore** is loaded by the device when it boots.

Common Operations

Data Manipulation

- `<get>`
- `<get-config>`
- `<edit-config>`
- `<copy-config>`
- `<delete-config>`
- `<discard-changes>` *(:candidate)*

Session Management

- `<close-session>`
- `<kill-session>`

Locking

- `<lock>`
- `<unlock>`

Transaction Management

- `<commit>` *(:candidate, :confirmed)*
- `<cancel-commit>` *(:candidate)*

Schema Management

- `<get-schema>` *(:monitoring)*

RPC Extensions

- `<rpc>`

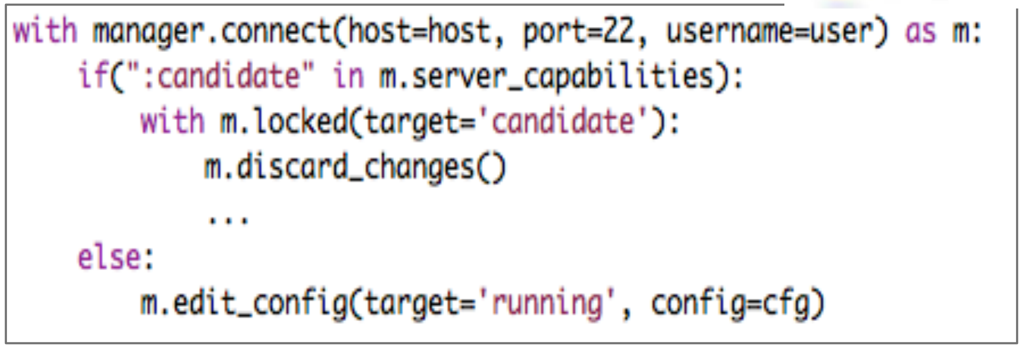
Anatomy of NETCONF Sessions

Ambitious version:

- **Hello** exchange including capabilities
- **Lock** running
- **Lock** candidate
- **Discard** changes on candidate
- **Edit** config on candidate
- **Commit** confirmed (with timeout)
- **Confirm** commit
- **Copy** running to startup
- **Unlock** candidate
- **Unlock** running

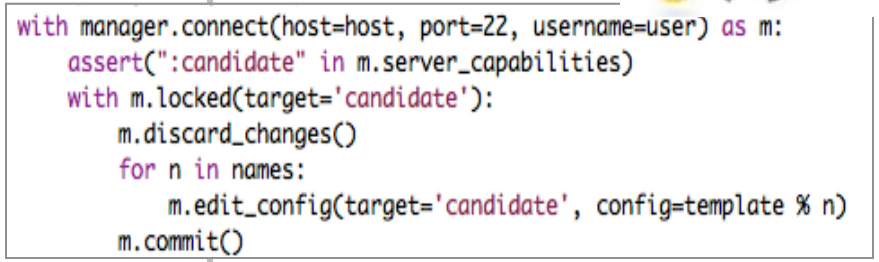
Short version:

- **Hello** exchange including capabilities
- **Edit** config on running database



```python
with manager.connect(host=host, port=22, username=user) as m:
    if(":candidate" in m.server_capabilities):
        with m.locked(target='candidate'):
            m.discard_changes()
            ...
    else:
        m.edit_config(target='running', config=cfg)
```

Example Exchange



```python
with manager.connect(host=host, port=22, username=user) as m:
    assert(":candidate" in m.server_capabilities)
    with m.locked(target='candidate'):
        m.discard_changes()
        for n in names:
            m.edit_config(target='candidate', config=template % n)
        m.commit()
```

```xml
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="5">
  <edit-config xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
    <target>
      <candidate/>
    </target>
    <test-option>test-then-set</test-option>
    <error-option>rollback-on-error</error-option>
    <config>
      <interface xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <name>eth1</name>
        <ipv4-address>192.168.5.10</ipv4-address>
        <macaddr>aa:bb:cc:dd:ee:ff</macaddr>
      </interface>
    </config>
  </edit-config>
</rpc>
```

```xml
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.1 message-id="5">
  <ok/>
</rpc-reply>
```

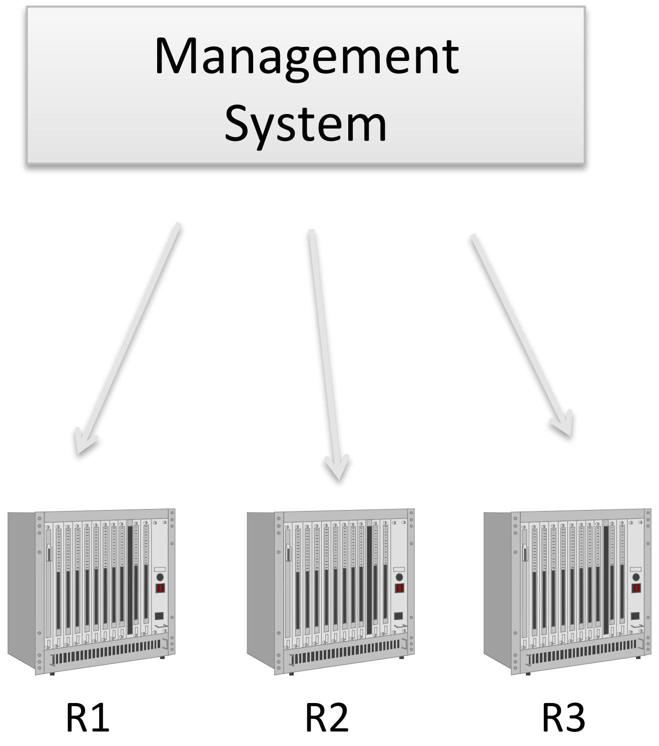
```xml
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.
  <validate>
    <source>
      <candidate/>
    </source>
  </validate>
</rpc>
```

```xml
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="6">
  <ok/>
</rpc-reply>
```

```xml
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.1
  <commit>
    <confirmed/>
  </commit>
</rpc>
```

```xml
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="7">
  <ok/>
</rpc-reply>
```

Distributed Transactions (for Bonus Points)



1. Connect to and lock R1, R2, R3
2. Edit candidate databases and commit with timeout
3. (Optionally) do assurance checks during timeout
4. Confirm commit, copy to startup and release locks

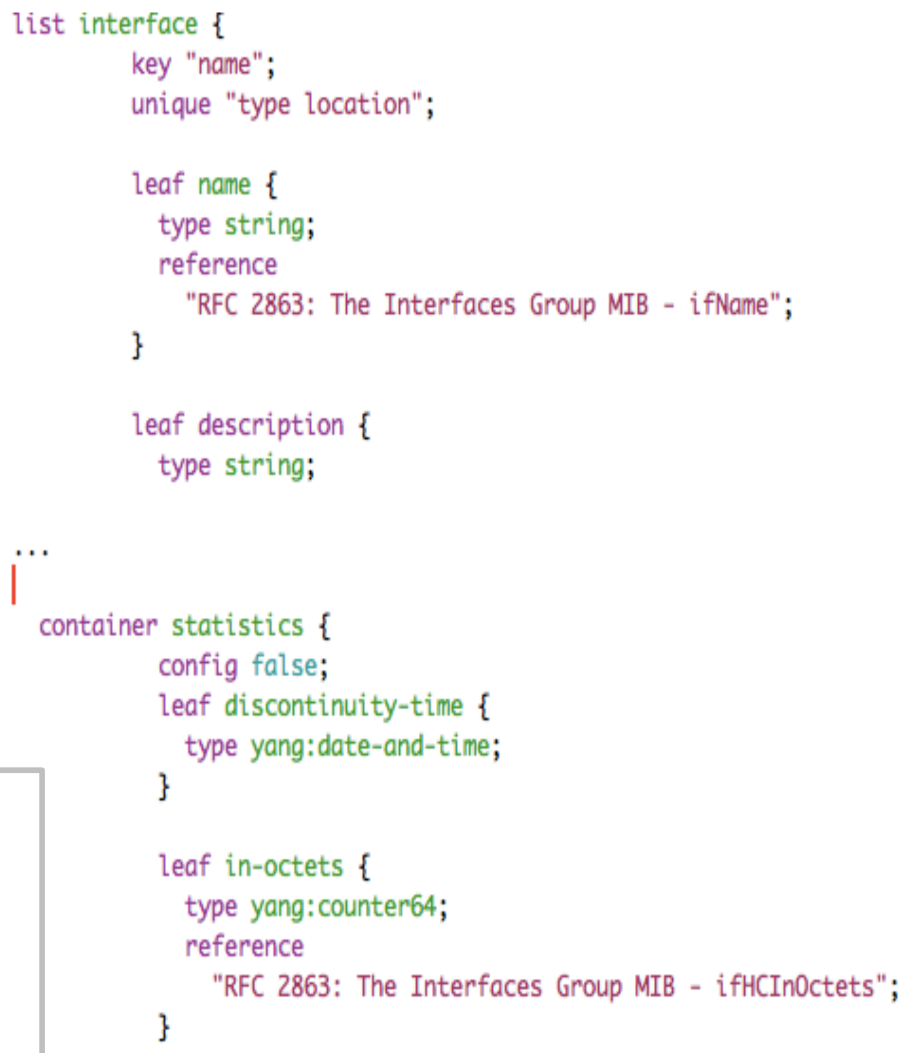
Transaction context manager simply kills all sessions on communication failure, failed commits -> Rollback

YANG – A Data Modeling Language for Networking

- Human readable, and easy to learn representation
- Hierarchical configuration data models
- Reusable types and groupings (structured types)
- Extensibility through augmentation mechanisms
- Supports definition of operations (RPCs)
- Formal constraints for configuration validation
- Data modularity through modules and sub-modules
- Well defined versioning rules

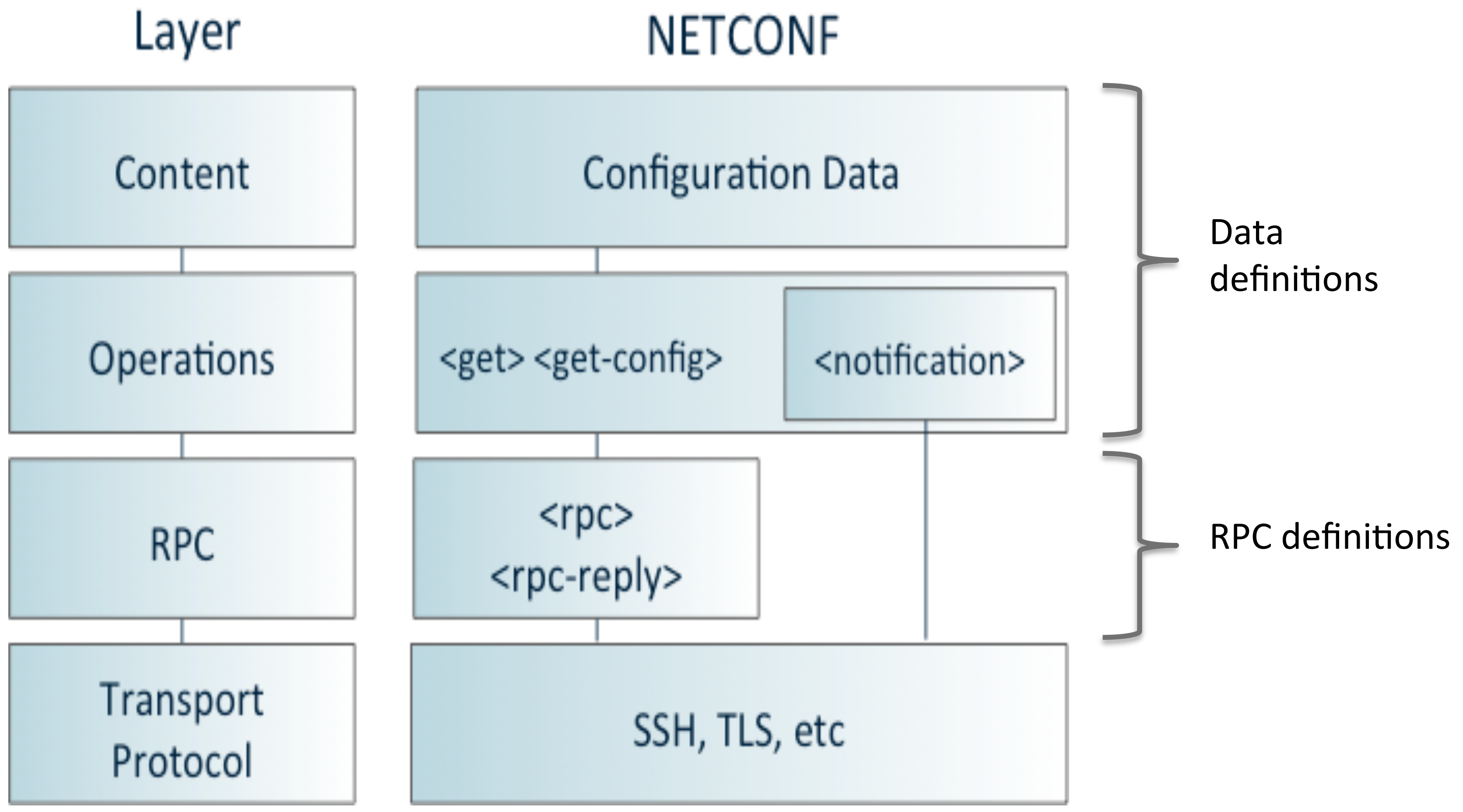
Why you should care:

YANG is a full, formal contract language with rich syntax and semantics to build applications on

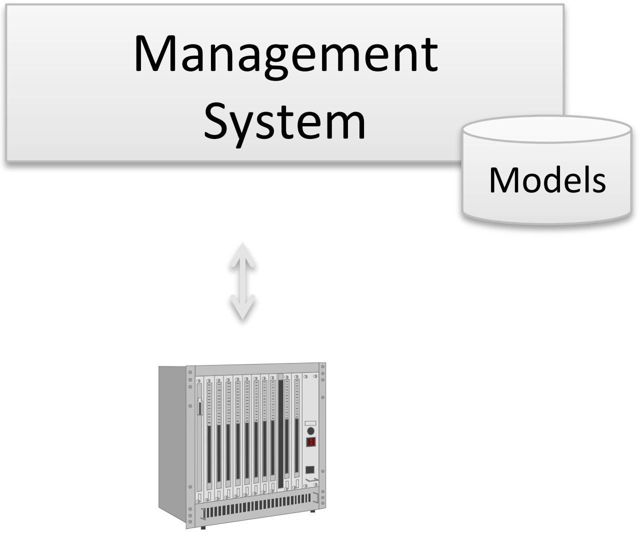


```
list interface {
        key "name";
        unique "type location";

        leaf name {
          type string;
          reference
            "RFC 2863: The Interfaces Group MIB - ifName";
        }

        leaf description {
          type string;

...

  container statistics {
          config false;
          leaf discontinuity-time {
            type yang:date-and-time;
          }

          leaf in-octets {
            type yang:counter64;
            reference
              "RFC 2863: The Interfaces Group MIB - ifHCInOctets";
          }
```

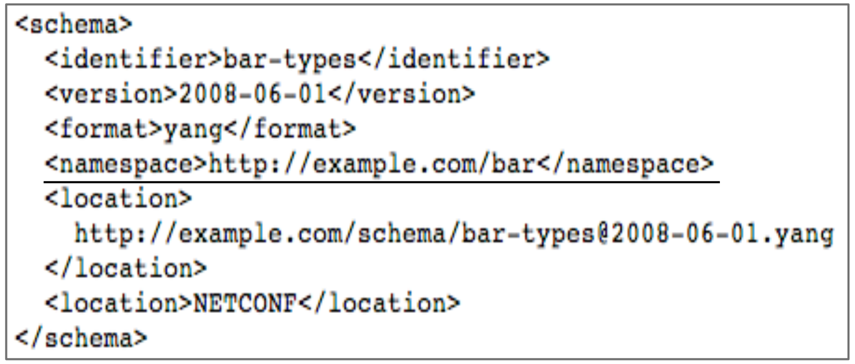
YANG in the NETCONF Layering Model



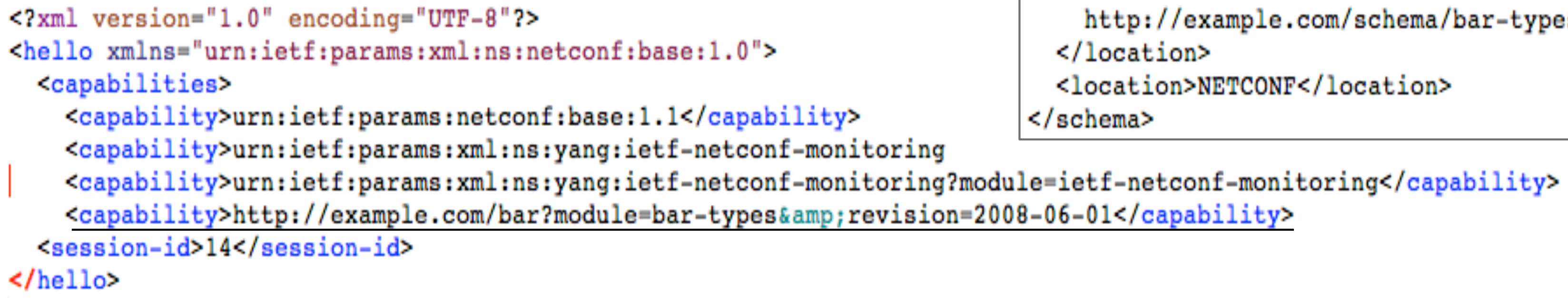
All Together Now



- Supported models, versions advertised through capabilities exchange
- Compare module identifiers with models in storage
- For each model that is not in storage:
  - Get-schema
- Update applications and drivers



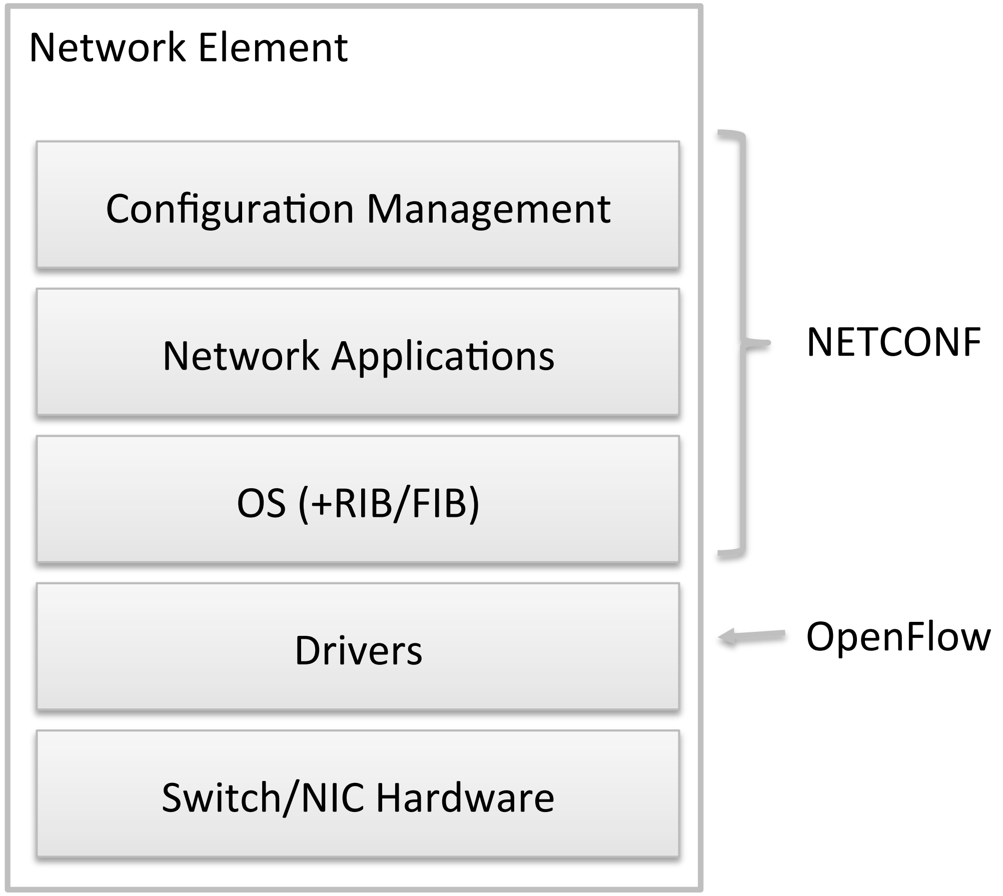
```
<schema>
  <identifier>bar-types</identifier>
  <version>2008-06-01</version>
  <format>yang</format>
  <namespace>http://example.com/bar</namespace>
  <location>
    http://example.com/schema/bar-types@2008-06-01.yang
  </location>
  <location>NETCONF</location>
</schema>
```



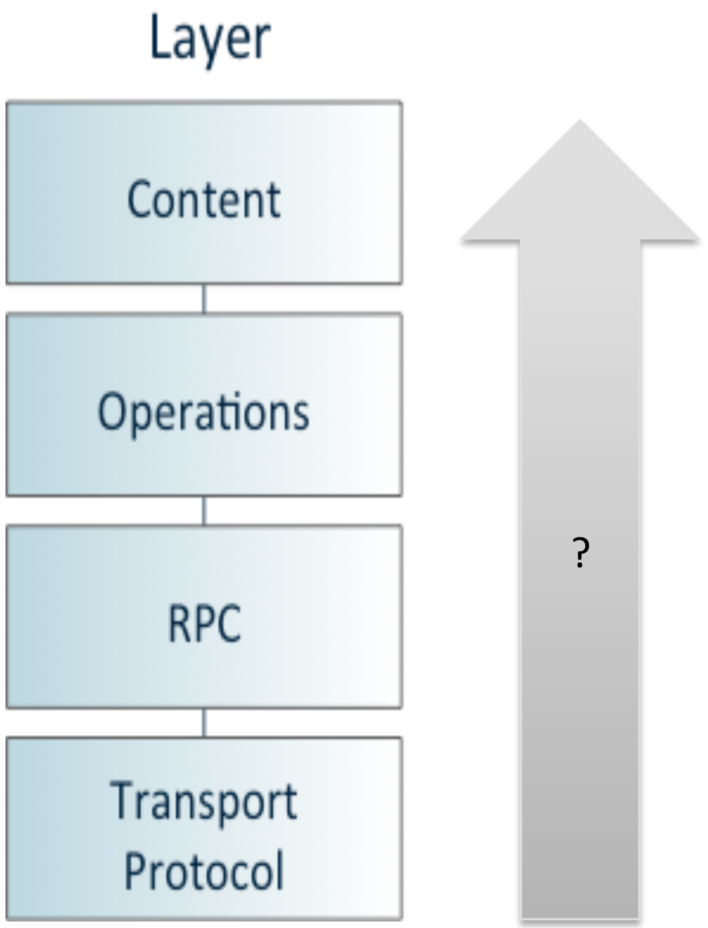
```
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.1</capability>
    <capability>urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring
    <capability>urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring?module=ietf-netconf-monitoring</capability>
    <capability>http://example.com/bar?module=bar-types&amp;revision=2008-06-01</capability>
  <session-id>14</session-id>
</hello>
```

Recommended Reading: RFC 6244 – NETCONF and YANG Architectural Overview

OpenFlow does that, right?



What About $PROTO? I Prefer $PROTO over NETCONF!



Momentum

A common, standard configuration management p data modeling language is needed in order t automation required by TeraStream. NETCONF and YANG are suitable because they are self-contained, in the sense that no additional offline knowledge about the syntax and transport of configuration exchange is needed. A CLI falls short since there is no formal data model and no formal vendor-neutral syntax. SNMP (Simple Network Management Protocol) falls short because it requires the application to have knowledge about in which order things can be created or modified.

DTAG TeraStream Device Management Interface Requirements

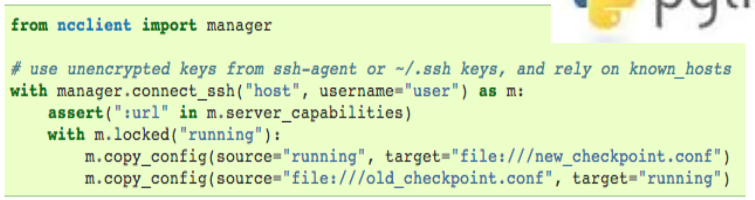
Deutsche Telekom

CISCO · ERICSSON · JUNIPER NETWORKS · BROCADE

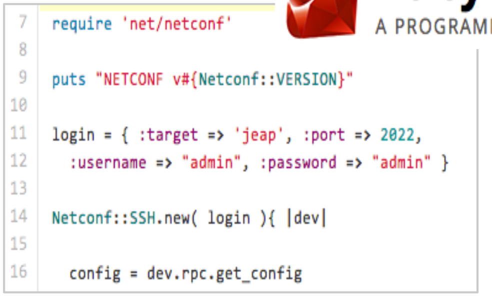
IETF · OPEN NETWORKING FOUNDATION · CableLabs · MEF METRO ETHERNET FORUM

tail-f · OPEN DAYLIGHT · MG SOFT · YumaWorks Automating the Internet

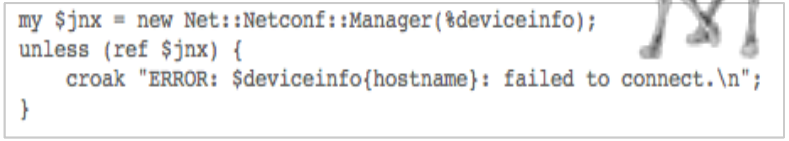
Language Bindings



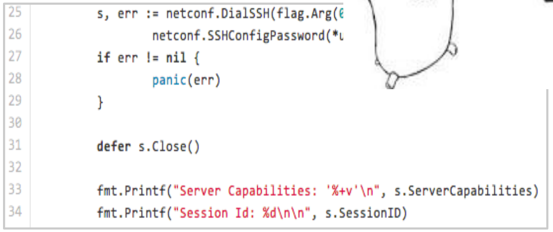
```python
from ncclient import manager

# use unencrypted keys from ssh-agent or ~/.ssh keys, and rely on known_hosts
with manager.connect_ssh("host", username="user") as m:
    assert(":url" in m.server_capabilities)
    with m.locked("running"):
        m.copy_config(source="running", target="file:///new_checkpoint.conf")
        m.copy_config(source="file:///old_checkpoint.conf", target="running")
```



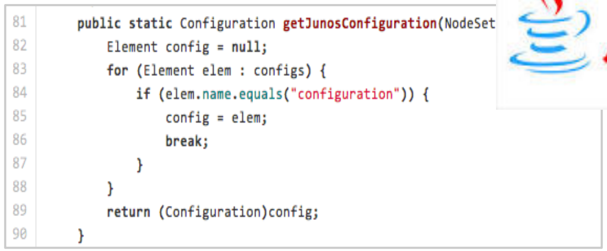
```ruby
require 'net/netconf'

puts "NETCONF v#{Netconf::VERSION}"

login = { :target => 'jeap', :port => 2022,
  :username => "admin", :password => "admin" }

Netconf::SSH.new( login ){ |dev|

  config = dev.rpc.get_config
```



```perl
my $jnx = new Net::Netconf::Manager(%deviceinfo);
unless (ref $jnx) {
    croak "ERROR: $deviceinfo{hostname}: failed to connect.\n";
}
```



```go
        s, err := netconf.DialSSH(flag.Arg(0
                netconf.SSHConfigPassword(*u
        if err != nil {
                panic(err)
        }

        defer s.Close()

        fmt.Printf("Server Capabilities: '%+v'\n", s.ServerCapabilities)
        fmt.Printf("Session Id: %d\n\n", s.SessionID)
```



```java
    public static Configuration getJunosConfiguration(NodeSet
        Element config = null;
        for (Element elem : configs) {
            if (elem.name.equals("configuration")) {
                config = elem;
                break;
            }
        }
        return (Configuration)config;
    }
```

IETF Activities

NETCONF Working Group

- *Rechartering in progress*
- NETCONF over TLS
- Reverse SSH

Maybe:

- RESTCONF
- DHCPv6 option for server discovery
- Efficiency extensions

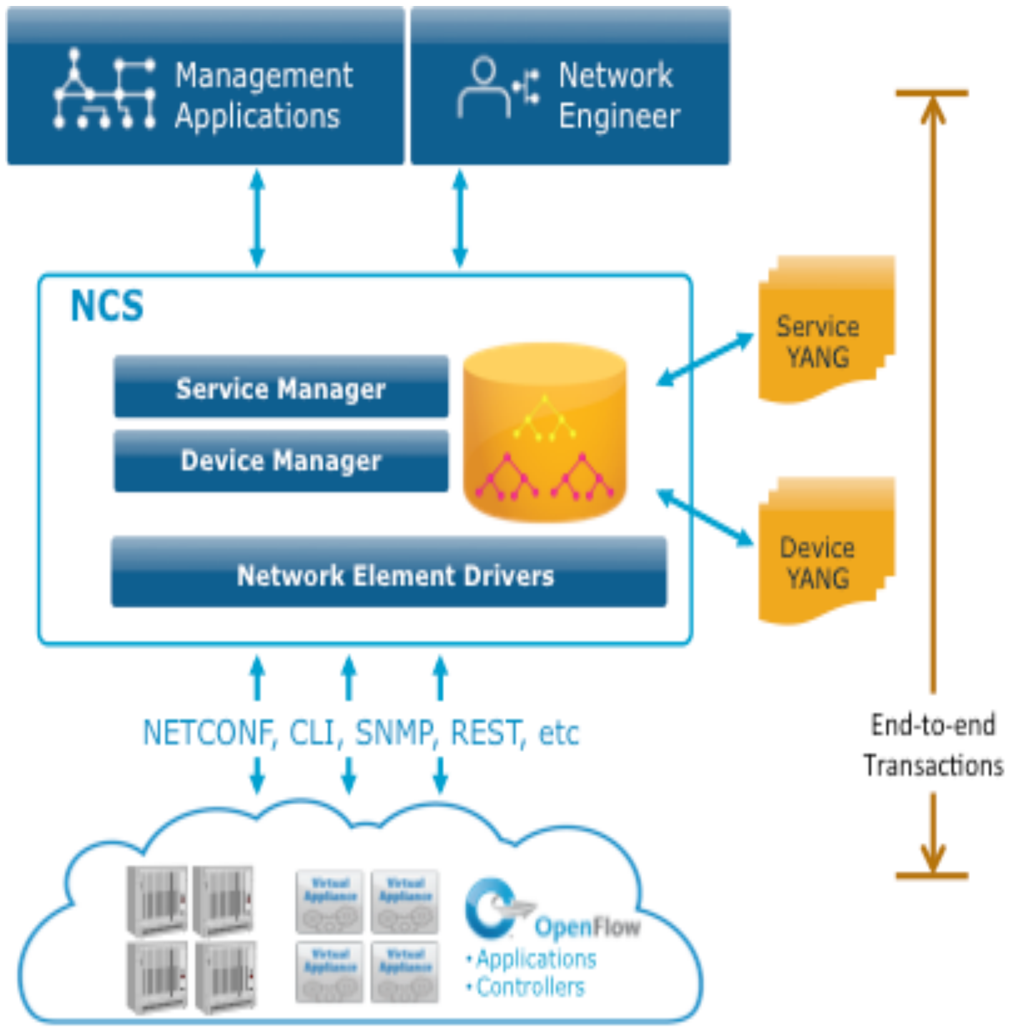
NETMOD Working Group

- Interface configuration
- IP address management
- Basic routing management
- System management (i.e. MIB-II)
- SNMP Configuration

Maybe:

- Topologies
- ACLs
- OSPF

And then you can build entire systems on it…



Questions and Further Reading

tail-f

calle@tail-f.com