Complexity Oblivious Network Management (CONMan)

Hitesh Ballani, Paul Francis

Cornell University

INM’06
Network Management is a Mess

- Ad-Hoc
- Complex
- Error-Prone
- Expensive

Worsening situation as network complexity increases

- 80% of IT budget in enterprises used to maintain status quo [Kerravala’04]
- Configuration errors account for 62% of network downtime [Kerravala’04]
Shortcomings of the existing architecture

- Dependency of the Management Plane on the Data Plane [4D, Greenberg et. al.’05]
- Control Plane Complexity [4D, Greenenberg et. al.’05]
  [RCP, Caesar et. al.’05]
- ...
Protocols expose their gory details

Hundreds of MIBs and Thousands of MIB objects

Get-info
Detailed Protocol and Device specific MIBs
Protocols expose their gory details

- Perception differs from reality
- Error-prone configuration
- Fragmentation of tools
Protocols expose their gory details

- Perception differs from reality
- Error-prone configuration
- Fragmentation of tools

Diagram:
- APPLICATIONS: UDP, TCP, GRE
- IP
- ATM, ETH, Frame Relay
- Management Applications
- Get-info
- Detailed Protocol and Device specific MIBs
- Approximate High-level Network Picture
Protocols expose their gory details

- Perception differs from reality
- Error-prone configuration
- Fragmentation of tools
Complexity Oblivious Network Management (CONMan)

A network management architecture that aims to

- Restrict protocol complexity to their implementation
Complexity Oblivious Network Management (CONMan)

A network management architecture that aims to

- Restrict protocol complexity to their implementation

Assumptions and Caveats

- Presence of an independent management channel
  [4D, Greenberg et. al.’05]
- “Network” management; not “Service” management
- Management of data-plane protocols
Restrict protocol details to implementation

Scenarios where details need not be exposed

- Key values for GRE tunnels
- Sequence numbers for GRE tunnels
- Filtering undesired packets
Restrict protocol details to implementation

Scenarios where details need not be exposed

- Key values for GRE tunnels
- Sequence numbers for GRE tunnels
- Filtering undesired packets

```
ip tun add name A mode gre remote 12.8.2.2 local\12.8.2.1 ikey 200 okey 1001 icsum ocsum iseq oseq
```
Restrict protocol details to implementation

Scenarios where details need not be exposed

- Key values for GRE tunnels
- Sequence numbers for GRE tunnels
- Filtering undesired packets

```
ip tun add name A mode gre remote 12.8.2.2 local\12.8.2.1 ikey 200 okey 1001 icsum ocsum iseq oseq
```
Restrict protocol details to implementation

Scenarios where details need not be exposed

- Key values for GRE tunnels
- Sequence numbers for GRE tunnels
- Filtering undesired packets

![Diagram of network setup with GRE tunnel and key values](image)

```bash
ip tun add name A mode gre remote 12.8.2.2 local 12.8.2.1 ikey 200 okey 1001 icsum ocsum iseq oseq

[Low Jitter/Delay] Vs [In-Order delivery]

Seq. No. Usage
```
Restrict protocol details to implementation

Scenarios where details need not be exposed

- Key values for GRE tunnels
- Sequence numbers for GRE tunnels
- Filtering undesired packets

"Filter packets from source address 128.19.2.3 and destined to address 20.3.4.5, port 592"
Abstract away the details

Protocols should not expose their gory details

What do the protocols expose?
Abstract away the details

What are these protocol modules doing?

What are these protocols modules doing?
Abstract away the details

What are these protocols modules doing?

**Switching** packets under some **performance** constraints while **filtering** unwanted traffic
Abstract away the details

What are these protocols modules doing?

Modules may depend on other modules for doing their job
Abstract away the details

Abstraction models the capabilities and dependencies of modules
Abstract away the details

Abstraction applies to (almost) all data plane modules
Abstract away the details

Abstraction applies to (almost) all data plane modules
CONMan Abstraction and Primitives

### Abstraction Components
- Name
- Up Pipes
- Down Pipes
- Physical Pipes
- Filter
- Switch
- Perf. Reporting
- Perf. Trade-off
- Security

### CONMan primitives
- show
- create
- conveyMessage
- test
Exceptions to the abstraction

Protocol details that need to be exposed

- IP address assignment
- Filtering based on regular expressions in HTML
- Broadcast suppression on switch ports
An example scenario: GRE Tunneling

Configuration at Router A
"Today"

```bash
#!/bin/bash
# Inserting the GRE-IP kernel module
insmod /lib/modules/2.6.10-1/ip_gre.ko
# Creating the GRE module with the appropriate key
ip tunnel add name greA mode remote 128.84.223.112 local 128.84.222.111 ikey 2001 okey 1001 icsum ocsum iseq oseq
ifconfig greA 192.168.1.3
# Enable routing
echo 1 > /proc/sys/net/ipv4/ip-forward
# Create IP routing state from customer to tunnel
echo 202 tun-1-2 > /etc/iproute2/rt_tables
ip rule add iff eth0 table tun-1-2 ip route add default dev greA table tun-1-2
# Create IP routing state from tunnel to customer
echo 203 tun-2-1 > /etc/iproute2/rt_tables
ip rule add iff greA table tun-2-1
ip route add default dev eth0 table tun-2-1
```
An example scenario: GRE Tunneling

```
#!/bin/bash
# Inserting the GRE-IP kernel module
insmod /lib/modules/2.6.10-1/ip_gre.ko
# Creating the GRE module with the appropriate key
ip tunnel add name greA mode remote 128.84.223.112 local 128.84.222.111 ikey 2001 okey 1001 icsum ocsum iseq oseq
ifconfig greA 192.168.1.3
# Enable routing
echo 1 > /proc/sys/net/ipv4/ip-forward
# Create IP routing state from customer to tunnel
echo 202 tun-1-2 > /etc/iproute2/rt_tables
ip rule add iff eth0 table tun-1-2 ip route add default dev greA table tun-1-2
# Create IP routing state from tunnel to customer
echo 203 tun-2-1 > /etc/iproute2/rt_tables
ip rule add iff greA table tun-2-1
ip route add default dev eth0 table tun-2-1
```
An example scenario: GRE Tunneling

**CONMan Goal:** “Create virtual connectivity between the customer-side interfaces for Customer-1”
NM discovers routers through the management channel

Uses *show* to determine the abstraction for the modules
An example scenario: GRE Tunneling

Map the high-level goal to the construction of path labeled (1) through (11)
An example scenario: GRE Tunneling

create (pipe, e, a)
create (pipe, a, d)
create (switch-state, a, pipe-2, pipe-3)
create (pipe, d, b)
create (pipe, b, c)

Configuration at Router A with CONMan
An example scenario: GRE Tunneling

GRE Modules use `conveyMessage` to exchange key values, seq numbers, etc.

```
create (pipe, e, a)
create (pipe, a, d)
create (switch-state, a, pipe-2, pipe-3)
create (pipe, d, b)
create (pipe, b, c)
```
An example scenario: GRE Tunneling

Configuration at Router A with CONMan

create (pipe, e, a)
create (pipe, a, d)
create (switch-state, a, pipe-2, pipe-3)
create (pipe, d, b)
create (pipe, b, c)

IP modules use conveyMessage to exchange and test IP addresses
Conclusion

- **CONMan**: a coherent network management architecture
- Moves operational complexity of protocols to their implementation

Protocols and devices modelled

- GRE protocol (tunnel configuration)
- IP protocol (performance management)
- Layer-2 switches (VLANs, VLAN tunneling, etc.)
Work in progress

Open Issues

- Evaluation strategies
- Scalability, performance and reliability issues
- Impact on security
- Deployment strategies
- ...
Work in progress

Open Issues

- Evaluation strategies
- Scalability, performance and reliability issues
- Impact on security
- Deployment strategies
- ...

Thank You!