Software Defined Networks:
Overview

CS6393
Lecture 9-1
Traditional Networks
Overview of Traditional Networks Architecture

Management Layer

Infrastructure Layer

Traditional Network

Network engineer

Command Line Interface (CLI)

Configuration commands

Control plane

Data plane

© Abdullah Al-Alaj
Network Devices in Traditional Networks

• Vendor-specific and heterogeneous.
• Requires manually configuration.
  • Costly.
  • High-rate of configuration errors.
  • Serious security breaches even for well-known security guidelines.
• Hardwired with specific algorithms and protocols to route, control and monitor data flow.
Problems in Traditional Networks

- Lack of global visibility of the network state.
- Difficulties in deploying and maintaining coherent network-wide policies.
- Innovation in networking functionalities and control is difficult.
- Unmanageable and high operational cost by network engineers.
- Complex and weak integration of decentralized networking devices.
- Hard to maintain stable and robust network security.
Security Policies in Traditional Networks

- Traditional networks lack security automation and run-time deployment of security policies.
- Lack for runtime update of security policies in response to traffic behavior or intrusions.
- To implement security policies, network operators need to
  - Translate high-level security policies into low-level configuration commands.
  - Implement low-level commands manually into large sets of vendor-specific devices.
- Updating security policies might require changing the hardware or updating its firmware.
Software Defined Networks
• SDN decouples the network control from the data forwarding plane in routers and switches.

• The result:
  • Control plane (logically centralized).
  • Forwarding plane takes decisions from the control plane.
  • Applications and services are implemented on top of the control Plane.
    • Control plan maps the entire network to the application layer.
  • Programmability of network functions and control.
Decoupling: The Idea behind SDNs

- **OpenFlow**
  - A Protocol between the control plane and data plane.
  - Describes how controller and a network forwarding device should communicate.

Software Defined Networks

- OpenFlow Controller
  - (Centralized) Control plane
  - Smart

- OpenFlow-enabled device
  - (Distributed) Data plane
  - Dumb, fast

© Abdullah Al-Alaj
Network Programmability

Abstract view

Northbound APIs
Network Programming Interface

Traffic mgmt.

Southbound Protocol (OpenFlow)

Application plane

Control plane

Data plane
The three Planes of SDN Architecture

- **Application Plane:** *SDN applications for various* functionalities, such as
  - network management
  - traffic automation
  - network Monitoring
  - security services.
  - etc.

- **Control Plane:** *Logically centralized control framework that:*
  - runs the Network Operating System (NOS)
  - maintains global view of the network.
  - provides hardware abstractions to SDN applications.

- **Data Plane:** *Forwarding elements* that:
  - forward traffic flows based on instructions from the control plane.
Overview of SDN Architecture

Applications
- Routing
- Firewall
- Load Balancing
- Intrusion Prevention
- Network Visualization
- Other

Network APIs
- Open Interface: needs control

Controller
- Network Services
  - Topology Service
  - Routing Service
  - Statistics Collection
  - Switch Management
  - Entry Pushing
  - Device Management
  - Link Discovery
  - Other

Infrastructure
- OpenFlow Protocol
- Flow table

Virtual Network Resources
- Topology
- Flow tables
- Switches
- Ports
- Statistics
- Traffic payloads
- Configurations
- VLANs

Control plane
- Decoupling

Data plane
- Hosts
- Links
- Devices
- Other
Enhanced Network Security in SDN

- Network security is enhanced in SDN via:
  - Network programmability.
  - Centralized control of network behavior.
  - Global visibility of the network state.
    - Easier to spot network vulnerabilities and intrusions.
    - Easier to implement security policies.
    - Easier to mitigate the risks of policy collision.
  - Run-time implementation of security policies.
  - Run-time manipulation of traffic forwarding rules.
  - Software based implementation of security policy.
    - No hardware change or firmware update needed.
# Flow Table Structure

<table>
<thead>
<tr>
<th>Match Fields</th>
<th>actions</th>
<th>priority</th>
<th>Counters</th>
<th>Timeouts</th>
<th>...</th>
</tr>
</thead>
</table>

- **match fields**: to match against packets.
- **actions**: to be applied for the matching packet/flow.
- **priority**: precedence of the flow entry.
- **counters**: updated when packets are matched.
- **timeouts**: maximum time/idle time before flow is expired by the switch.

## Packet Processing in OpenFlow Switch

1. **Packet from network**
2. **Parse header fields**
3. **Match Found?**
   - Yes: **Update counters**
   - No: **Send to controller**
4. **Apply actions**
Flow Table Structure
example flow rules

OpenFlow Table entry

<table>
<thead>
<tr>
<th>Rule</th>
<th>Priority</th>
<th>Action</th>
<th>Counters</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC src</td>
<td>MAC dst</td>
<td>IP src</td>
<td>IP dst</td>
</tr>
<tr>
<td>..:00:01</td>
<td>..:00:04</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>..:00:04</td>
<td>..:00:01</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>10.0.0.3</td>
</tr>
</tbody>
</table>

Packet Processing in OpenFlow Switch

1. Packet from network
2. Parse header fields
3. Match Found?
   - No: Send to controller
   - Yes: Update counters
4. Apply actions
Flow Rule Insertion Example - Routing app

1. Switch receives a network packet
2. Parse packet
3. Forward packet to network apps
4. If a matching rule found in table, apply actions; otherwise, forward packet to controller.
5. Read topology info.
6. Find shortest path
7. Read hosts info.
8. Insert flow rule

Flow rule: Mac-Host-A -> Mac-Host-D : Port-4

Packet Processing in OpenFlow Switch:
- Packet from network
- Parse header fields
- Match Found?
  - No: Send to controller
  - Yes: Update counters
    - Apply actions

© Abdullah Al-Alaj
Flow Rule Insertion Example - Firewall app

Security Policy:
- Deny connections from Host-A to Host-C.
- Allow connections from Host-B to Host-C.

Switch receives a network packet

1. Switch receives a network packet
2. Decision to controller
3. If no matching rule in switch, forward packet to controller
4. Parse packet
5. Check security policy
6. Enforce security policy
7. Decision to controller
8. Insert flow rule: Host-A -> Host-C: Drop

Forward packet to network apps

© Abdullah Al-Alaj
SDN Security Challenges
Apparent Control Plane Security Challenges

- Threats due to Scalability
  - Huge # flow rules

- DoS Attacks
  - SDN response times
  - IP packets with random headers

- Challenges in Distributed Control Plane
  - Information aggregation from multiple domains.
  - Different privacy rules in different domains.

Application Plane Security Challenges

- Lack of authentication mechanisms for SDN apps
  Requires a trust relationship between the controller and apps

- Lack of access control for SDN apps
  Requires a permission system to apply least privilege on apps

- Fraudulent and contradicting flow rule insertion
  Requires a flow rule conflict resolution

Assume this sequence of activities by SDN Apps.

1. **Security App** has identified Server-2 as Malicious server.
2. **Security App** inserts rule1 & rule2 to quarantine the flows to/from Server-2.
4. **Load Balancer App** inserts flow Rules rule3 & rule4 to redirect traffic to Server-2.

Load Balancer App violates the security policy implemented by Security App.

---

© Abdullah Al-Alaj