I DPID It My Way! A Covert Timing Channel in Software-Defined Networks

Robert Krösche, **Kashyap Thimmaraju**, Liron Schiff and Stefan Schmid

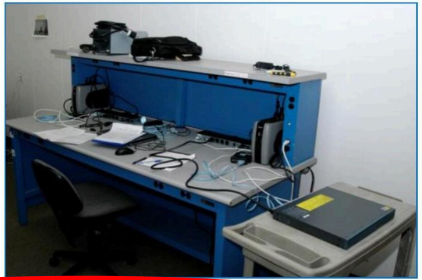
IFIP Networking 2018
14-16 May, 2018, Zurich, Switzerland

Outline

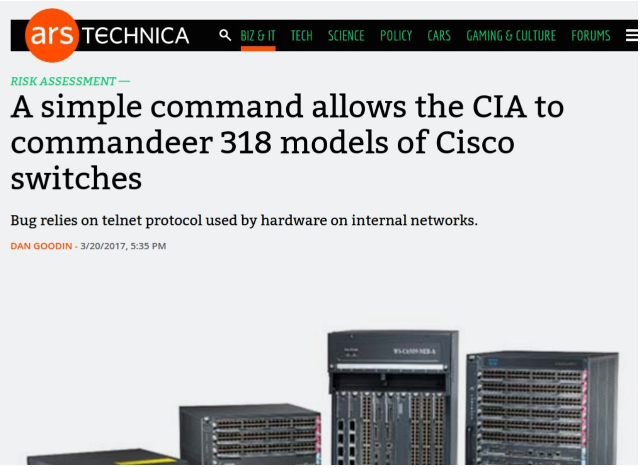
1. Motivation
2. Covert Timing Channel
3. CVE-2018-1000155
4. Conclusion

Backdoors and Exploits

(TS//SI//NF) Such operations involving **supply-chain interdiction** are some of the most productive operations in TAO, because they pre-position access points into hard target networks around the world.



(TS//SI//NF) Left: Intercepted packages are opened carefully; Right: A "load station" implants a beacon



RISK ASSESSMENT —

A simple command allows the CIA to commandeer 318 models of Cisco switches

Bug relies on telnet protocol used by hardware on internal networks.

DAN GOODIN - 3/20/2017, 5:35 PM

Also Possible With SDN (Virtual) Switches! [SOSR'18]

Taking Control of SDN-based Cloud Systems via the Data Plane

Kashyap Thimmaraju
Security in Telecommunications
TU Berlin
Berlin, Germany
kash@sect.tu-berlin.de

Bhargava Shastry
Security in Telecommunications
TU Berlin
Berlin, Germany
bshastry@sect.tu-berlin.de

Tobias Fiebig
Faculty of Technology, Policy and Management
TU Delft
Delft, Netherlands
t.fiebig@tudelft.nl

Felicitas Hetzelt
Security in Telecommunications
TU Berlin
Berlin, Germany
file@sect.tu-berlin.de

Jean-Pierre Seifert
Security in Telecommunications
TU Berlin
Berlin, Germany
jpseifert@sect.tu-berlin.de

Anja Feldmann
Internet Architecture
Max-Planck-Institut für Informatik
Saarbrücken, Germany
anja@mpi-inf.mpg.de

Stefan Schmid*†
Faculty of Computer Science
University of Vienna
Vienna, Austria
schmiste@univie.ac.at

ABSTRACT

Virtual switches are a crucial component of SDN-based cloud systems, enabling the interconnection of virtual machines in a flexible and "software-defined" manner. This paper raises the alarm on the security implications of virtual switches. In particular, we show that virtual switches not only *increase the attack surface* of the cloud, but virtual switch vulnerabilities can also lead to attacks of much *higher impact* compared to traditional switches.

We present a systematic security analysis and identify four design decisions which introduce vulnerabilities. Our findings motivate us to revisit existing threat models for SDN-based cloud setups, and introduce a new attacker model for SDN-based cloud systems using virtual switches.

We demonstrate the practical relevance of our analysis using a case study with Open vSwitch and OpenStack. Employing a fuzzing methodology, we find several exploitable vulnerabilities in Open vSwitch. Using just one vulnerability we were able to create a worm that can compromise hundreds of servers in a matter of minutes.

Our findings are applicable beyond virtual switches: NFV and high-performance fast path implementations face similar issues. This paper also studies various mitigation techniques and discusses how to redesign virtual switches for their integration.

KEYWORDS

Network Isolation; Network Virtualization; Data Plane Security; Packet Parsing; MPLS; Virtual Switches; Open vSwitch; Cloud Security; OpenStack; Attacker Models; ROP; SDN; NFV

1 INTRODUCTION

Modern cloud systems such as OpenStack [7], Microsoft Azure [26] and Google Cloud Platform [92] are designed for programmability, (logically) centralized network control and global visibility. These tenets also lie at the heart of Software-defined Networking (SDN) [23, 51] which enables cloud providers to efficiently utilize their resources [35], manage their multi-tenant networks [44], and reason about orchestration [41].

The data plane of Software-Defined Networks in the cloud are highly virtualized [44]: Virtual switches (running on

*Also with, Internet Network Architectures, TU Berlin.
†Also with, Dept. of Computer Science, Aalborg University.

SOSR '18, March 28–29, 2018, Los Angeles, CA, USA

ACM ISBN 978-1-4503-5664-0/18/03...$15.00

Malicious SDN Switches

SDN Teleportation [EuroSP'17]

A New Attack in
Software-Defined Networks

Outsmarting Network Security with SDN Teleportation

Kashyap Thimmaraju
Security in Telecommunications
TU Berlin
Berlin, Germany
Email: kash@fgsect.de

Liron Schiff
GuardiCore Labs
Tel Aviv, Israel
Email: liron.schiff@guardicore.com

Stefan Schmid
Dept. of Computer Science
Aalborg University
Aalborg, Denmark
Email: schmiste@cs.aau.dk

***Abstract*—Software-defined networking is considered a promising new paradigm, enabling more reliable and formally verifiable communication networks. However, this paper shows that the separation of the control plane from the data plane, which lies at the heart of Software-Defined Networks (SDNs), introduces a new vulnerability which we call *teleportation*. An attacker (e.g., a malicious switch in the data plane or a host connected to the network) can use teleportation to transmit information via the control plane and bypass critical network functions in the data plane (e.g., a firewall), and to violate security policies as well as logical and even physical separations. This paper characterizes the design space for teleportation attacks theoretically, and then identifies four different teleportation techniques. We demonstrate and discuss how these techniques can be exploited for different attacks (e.g., exfiltrating confidential data at high rates), and also initiate the discussion of possible countermeasures. Generally, and given today's trend toward more intent-based networking, we believe that our findings are relevant beyond the use cases considered in this paper.**

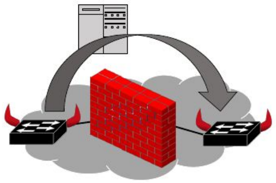
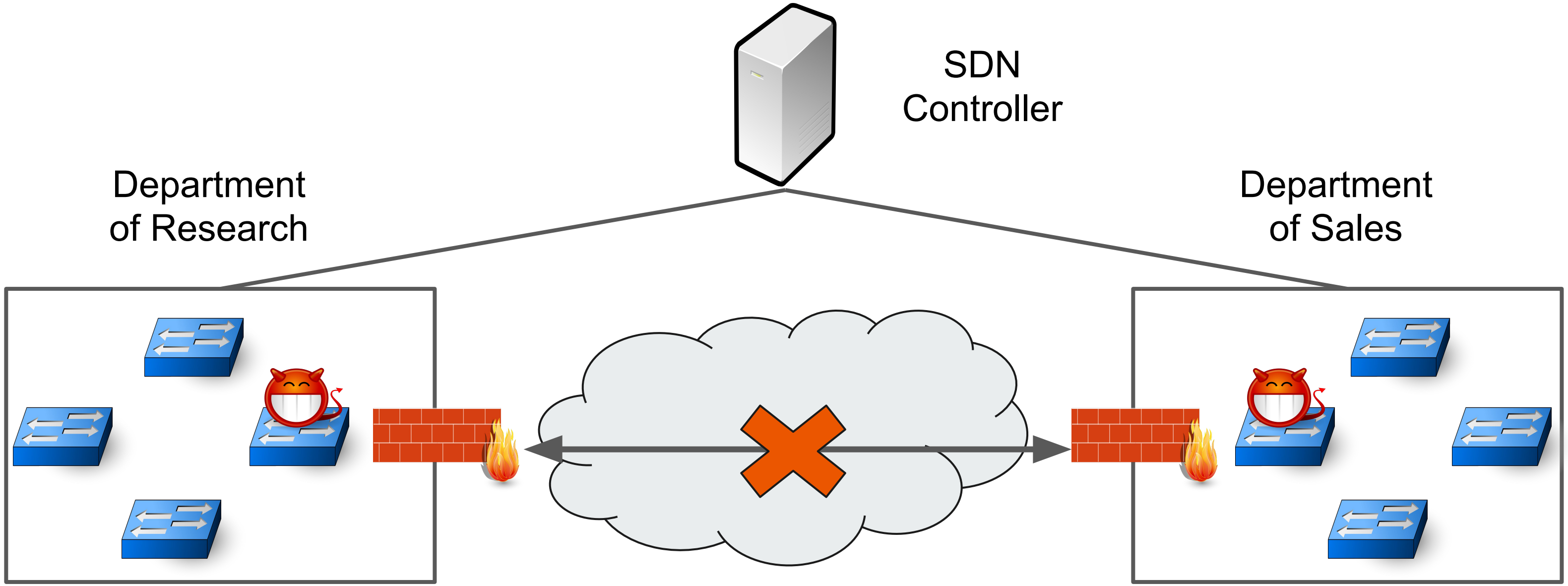


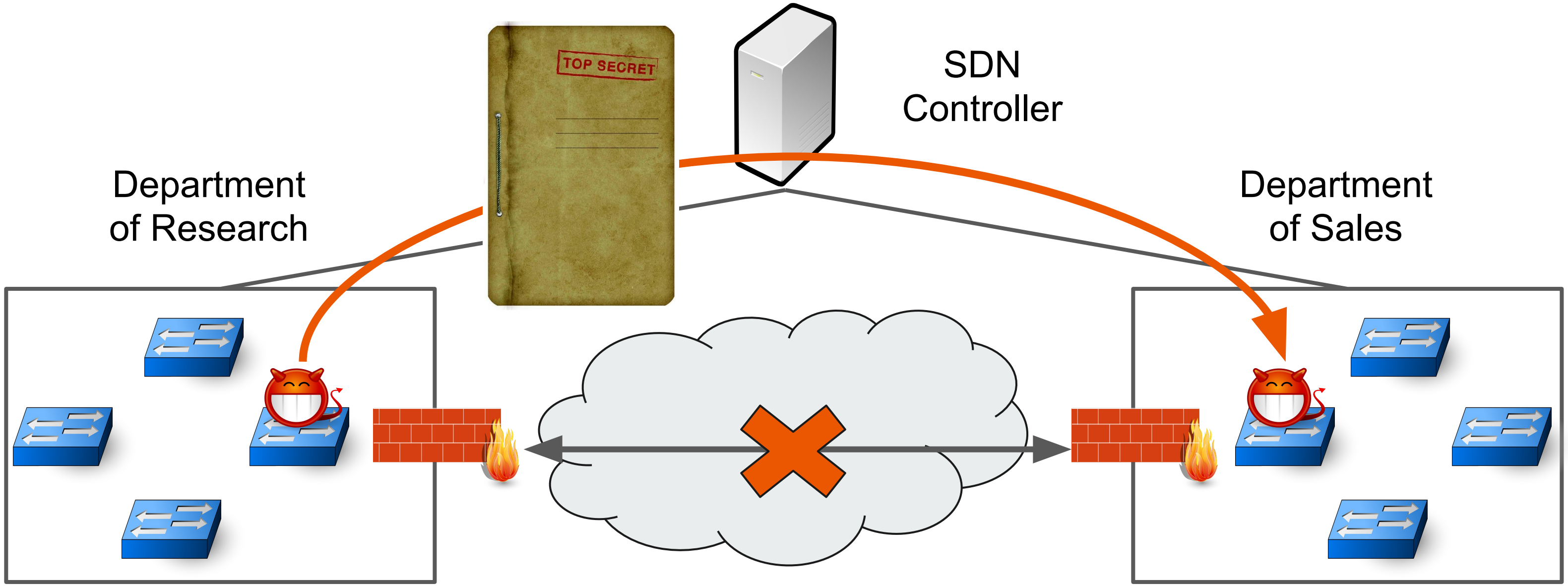
Figure 1: Illustration of teleportation: Malicious switches (with *red horns*) exploit the control platform for hidden communication, possibly bypassing data plane security mechanisms such as a firewall.

tions, also in terms of security, through its decoupling and consolidation of the control plane, its formally verifiable

SDN Teleportation: Violate Network Isolation

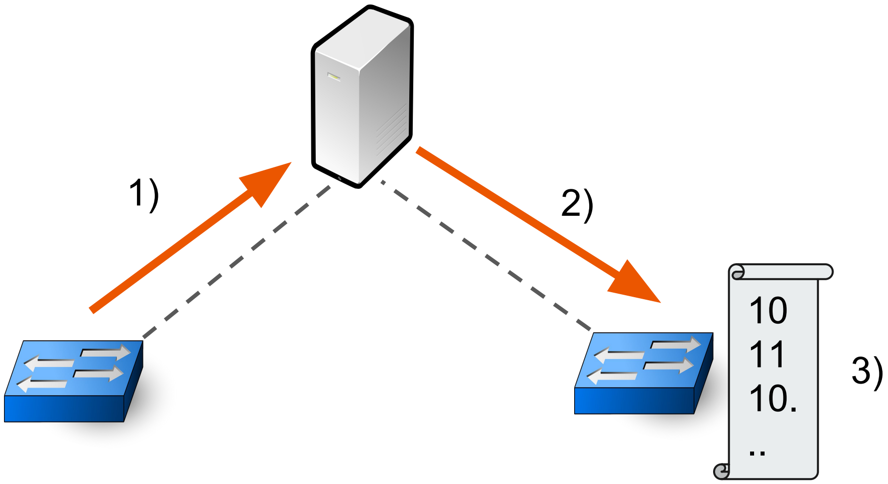


SDN Teleportation: Violate Network Isolation



The Teleportation Model

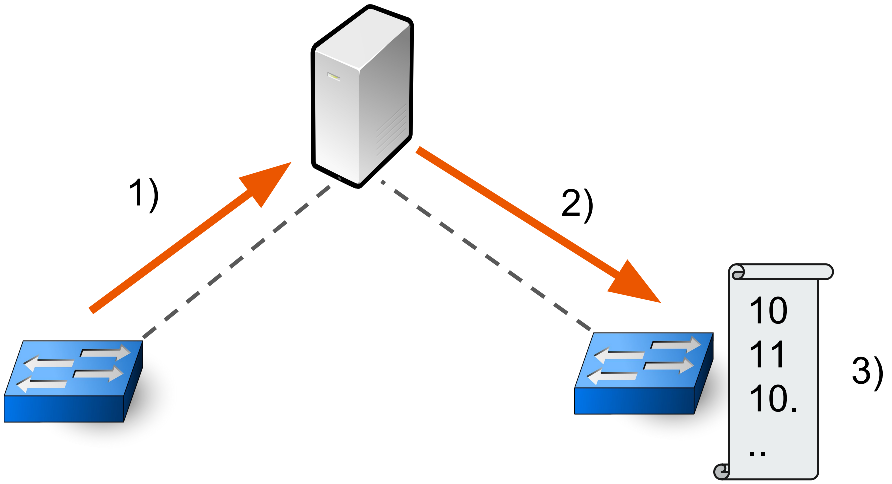
1) Switch to Controller
2) Controller to Switches
3) Destination Processing



Teleportation Techniques

- Out-of-band Forwarding
- Flow (Re-)Configuration
- **Switch Identification**

Inherent to the OpenFlow specification



Switch Identification Teleportation

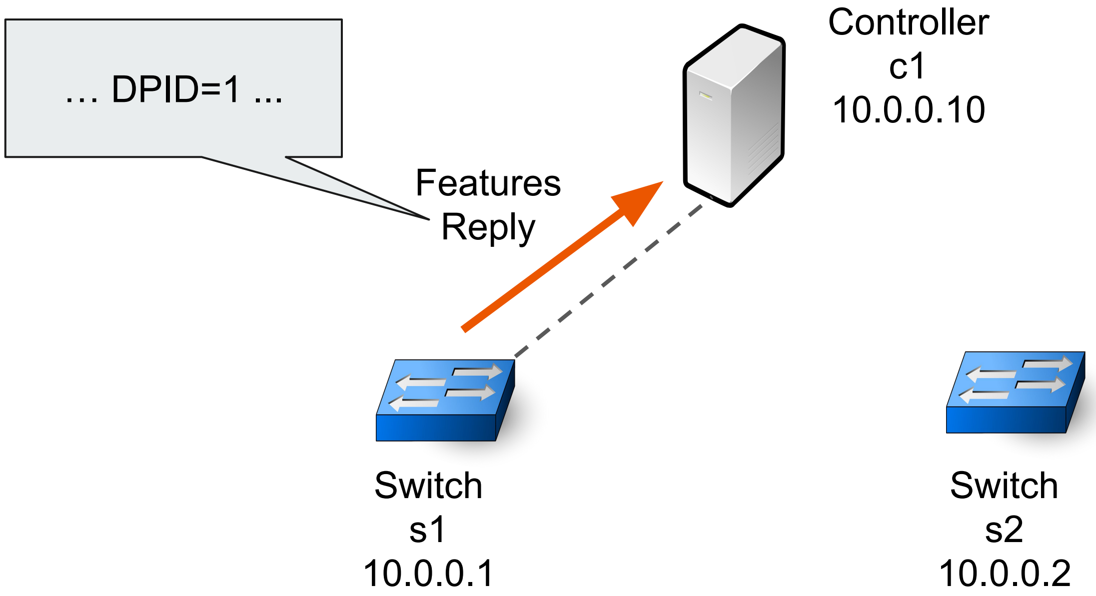
A Covert Timing Channel

- OpenFlow Handshake
- Switches use the same Data Path Identifier (DPID) to the same controller

Switch Identification Teleportation

OpenFlow Messages

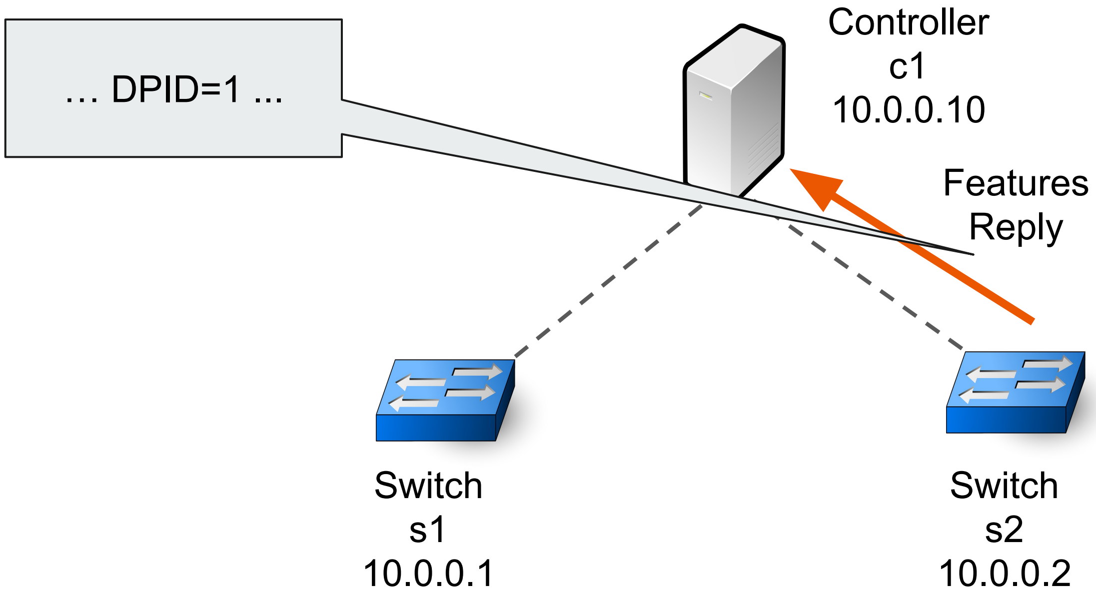
- Hello
- Features Request
- Features Reply



Switch Identification Teleportation

OpenFlow Messages

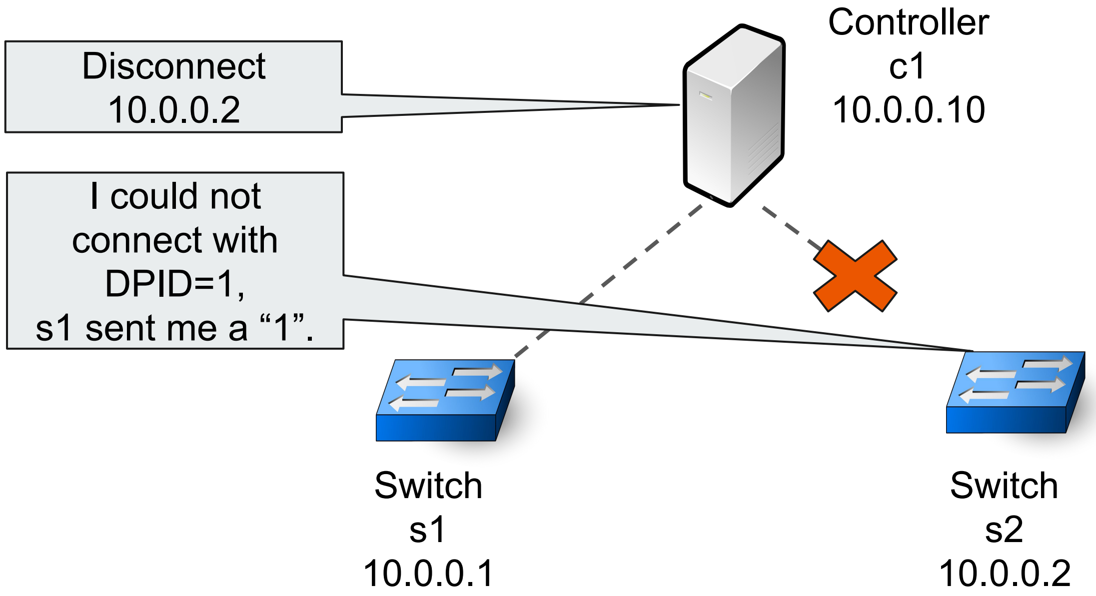
- Hello
- Features Request
- Features Reply



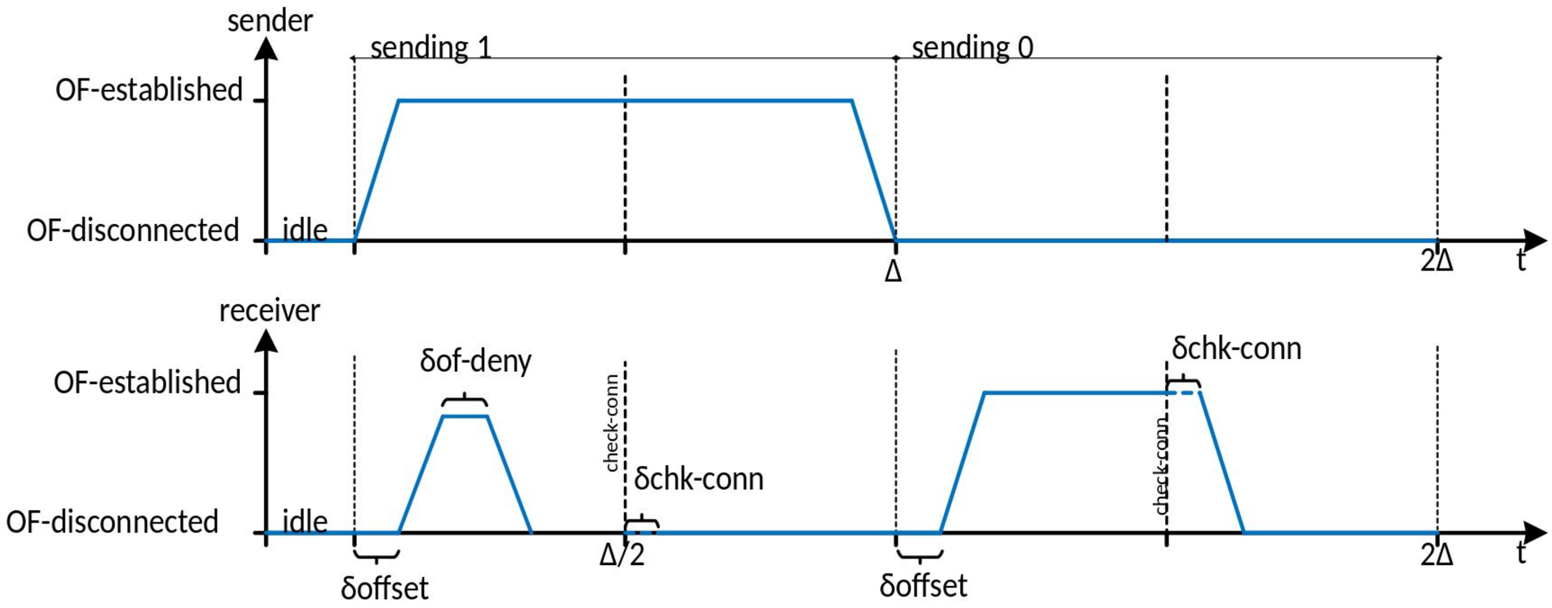
Switch Identification Teleportation

OpenFlow Messages

- Hello
- Features Request
- Features Reply

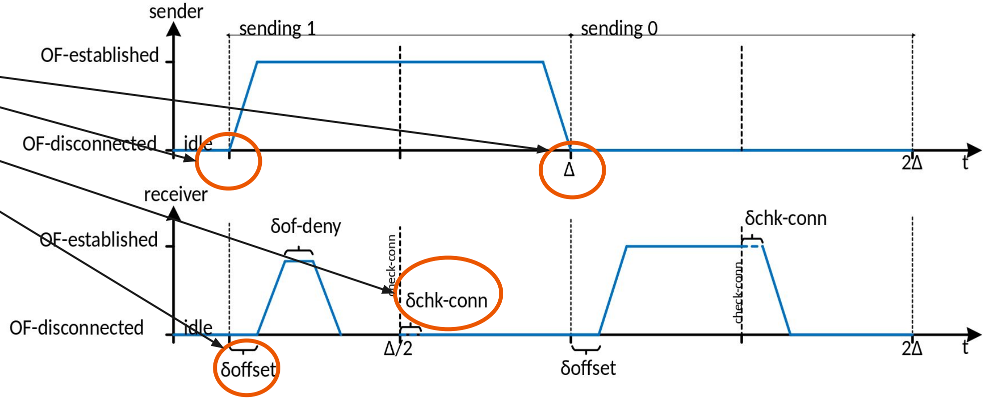


Covert Timing Channel



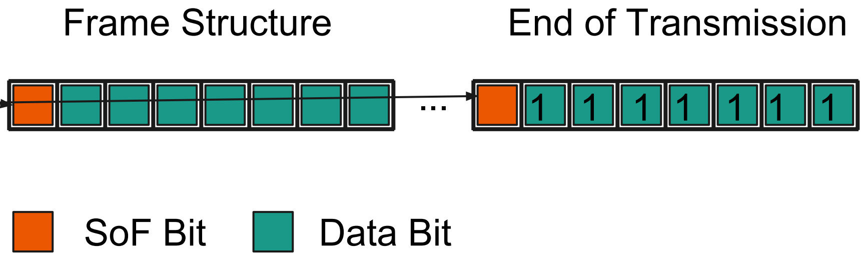
Challenges From One Bit to Multiple Bits

- Synchronization
  - When to start?
  - How long to wait?
  - Did it start?
  - When to end?
- Influence of the Controller
  - Load on the controller
  - Controller architecture
  - Path to the controller



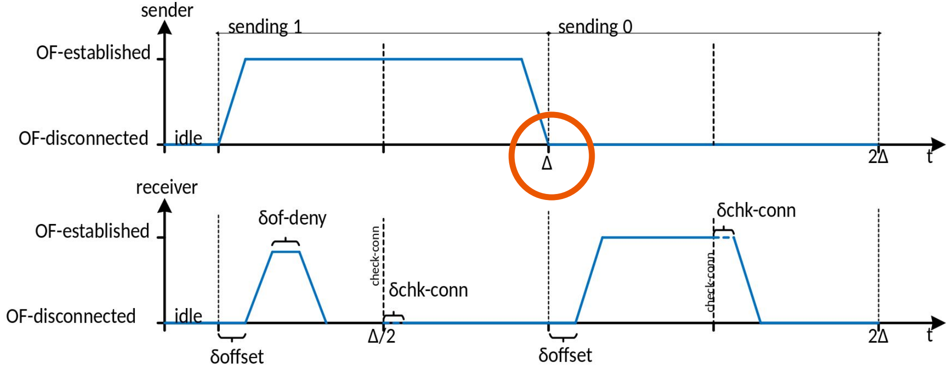
Challenges From One Bit to Multiple Bits

- Synchronization
  - When to start?
  - How long to wait?
  - Did it start?
  - When to end?
- Influence of the Controller
  - Load on the controller
  - Controller architecture
  - Path to the controller

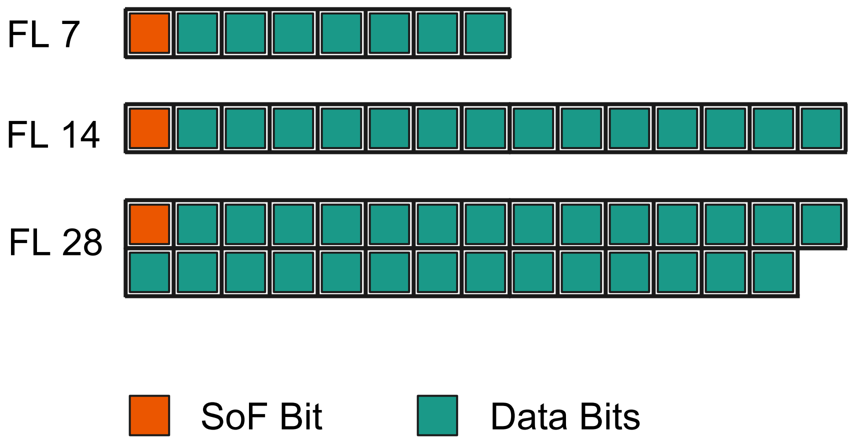


Experimental Evaluation

Effect of Timing Interval (Δ)

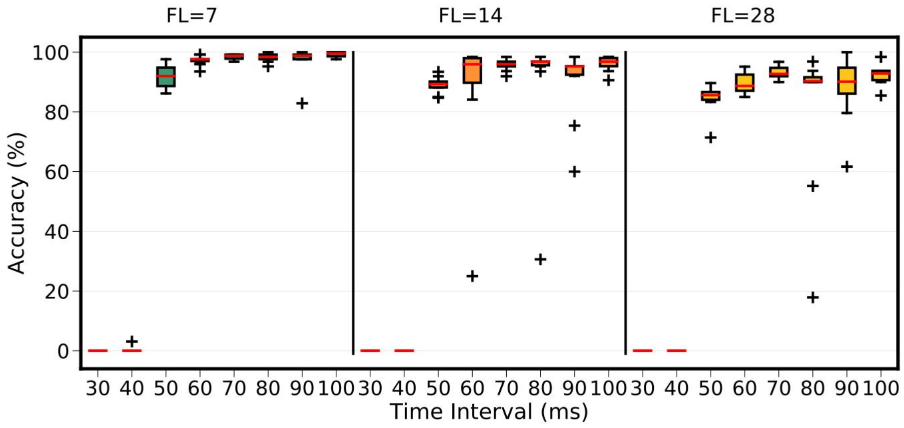


Effect of Frame Length (FL)



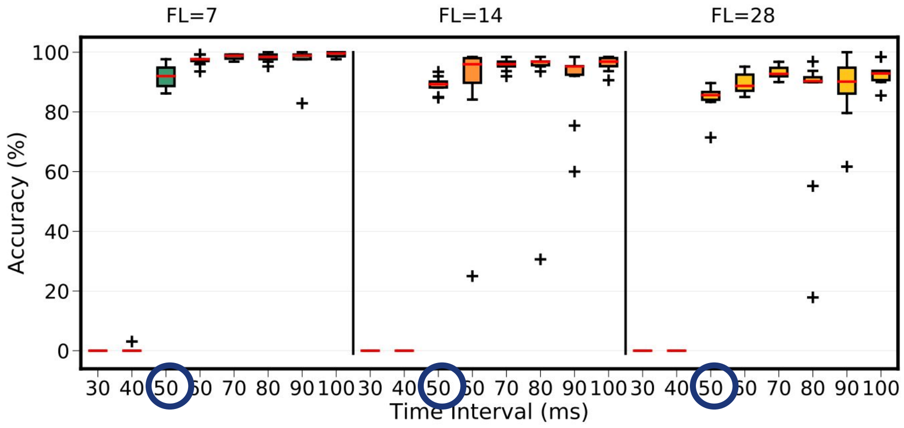
Effect of Timing Interval (Δ) and Frame Length (FL)

No load, M=64bytes, δ_{offset}=5ms and check the conn. status at Δ/2



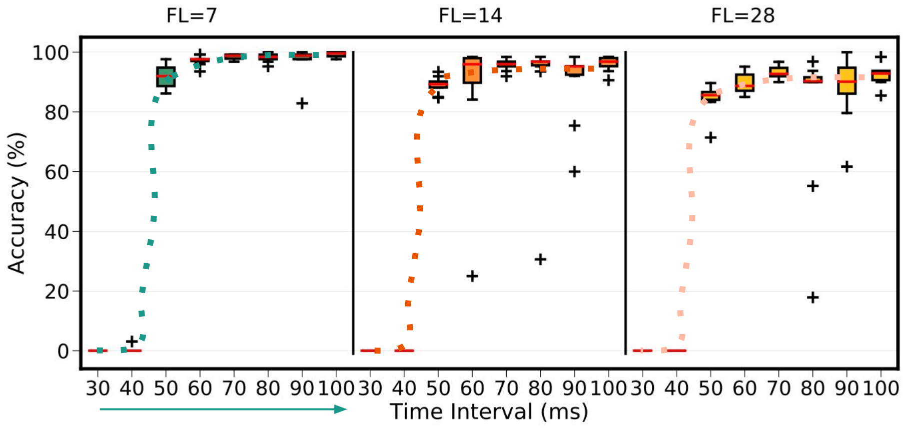
Effect of Timing Interval (Δ) and Frame Length (FL)

No load, M=64bytes, δ_{offset}=5ms and check the conn. status at Δ/2



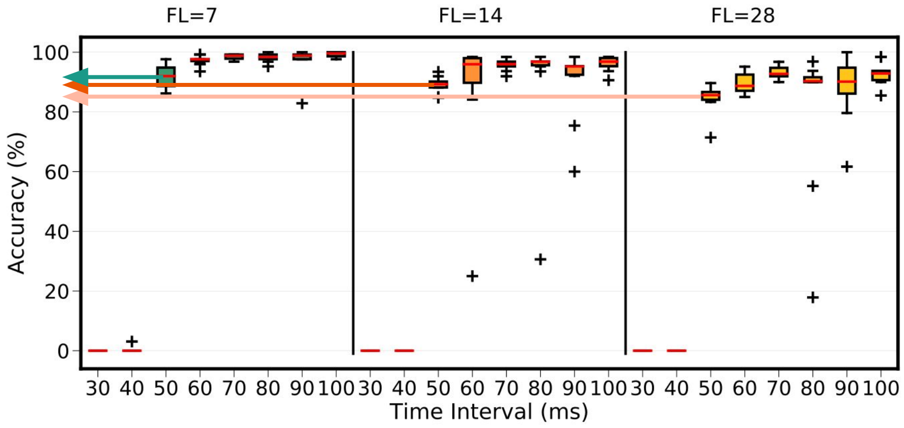
Effect of Timing Interval (Δ) and Frame Length (FL)

No load, M=64bytes, δ_{offset}=5ms and check the conn. status at Δ/2

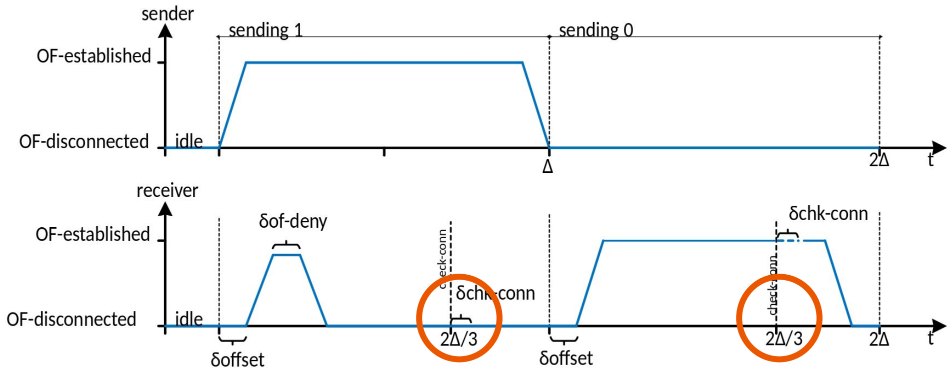


Effect of Timing Interval (Δ) and Frame Length (FL)

No load, M=64bytes, δ_{offset}=5ms and check the conn. status at Δ/2

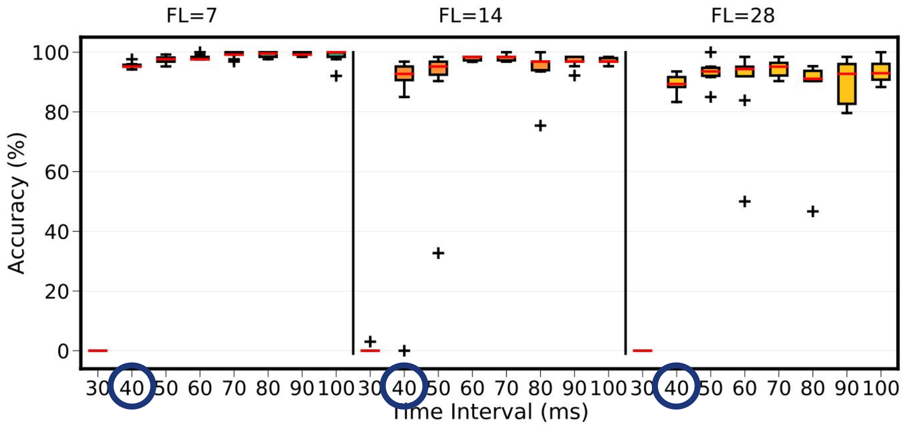


Effect of Delay (δ_{delay}) to Check Conn. Status

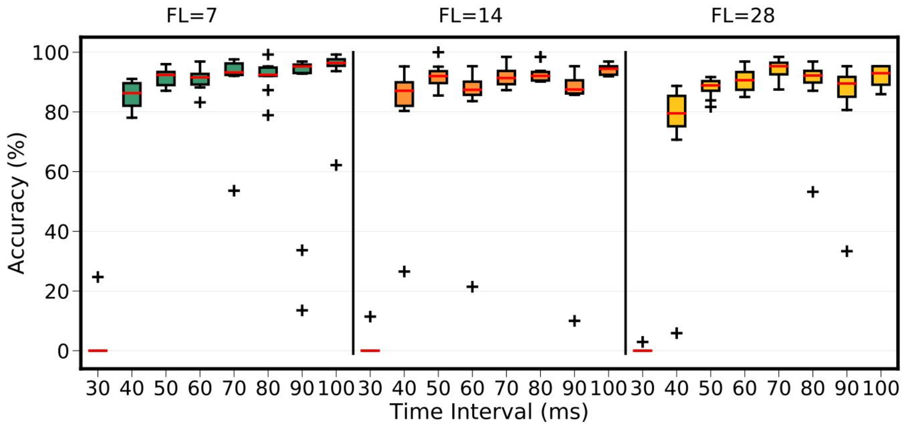


Effect of Delay (δ_{delay}) to Check Conn. Status

No load, M=64bytes, δ_{offset}=5ms and check the conn. status at 2Δ/3



Effect of Load on the Controller



With load (20 switches trigger Packet-Ins following a Poisson distribution with λ=1), M=64bytes, δ_{offset}=5ms and check the conn. status at $2\Delta/3$

Limitations, Detection and Mitigation

- Uni-directional and no error-correction in our prototype
- System and network limitations, e.g., TCP connection establishment time
- It is difficult to detect Teleportation attacks as the (OpenFlow) messages are legitimate and within the switch-controller channel
- We can deter Switch Identification Teleportation by securing the OpenFlow handshake

CVE-2018-1000155

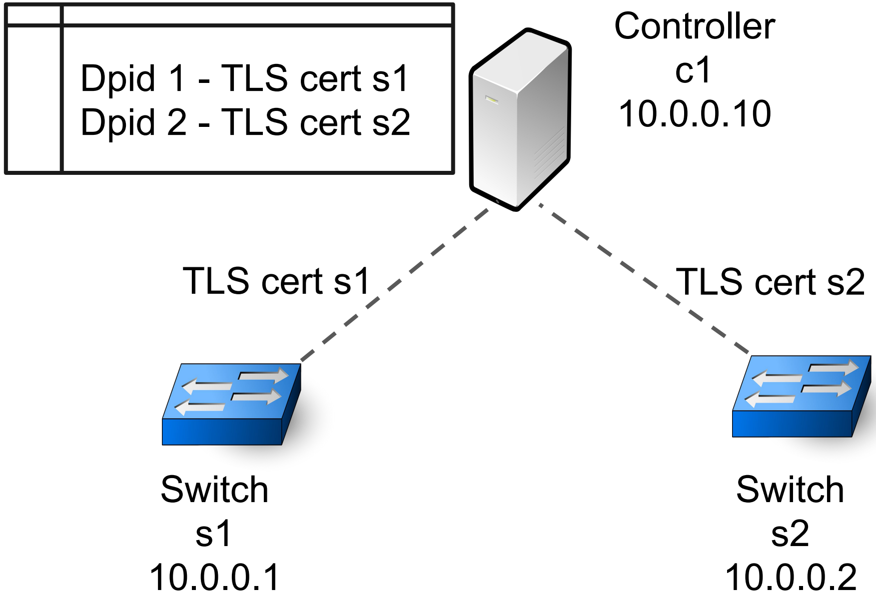
- Lack of authentication
- Lack of authorization
- Denial of service
- Difficult to specify the outcome for a switch ID collision at the controller in OpenFlow

- Public disclosure made last week
  - http://www.openwall.com/lists/oss-security/2018/05/09/4
  - https://www.theregister.co.uk/2018/05/10/openflow_switch_auth_vulnerability/
  - https://www.techrepublic.com/article/openflow-sdn-protocol-flaw-affects-all-versions-could-lead-to-dos-attack/

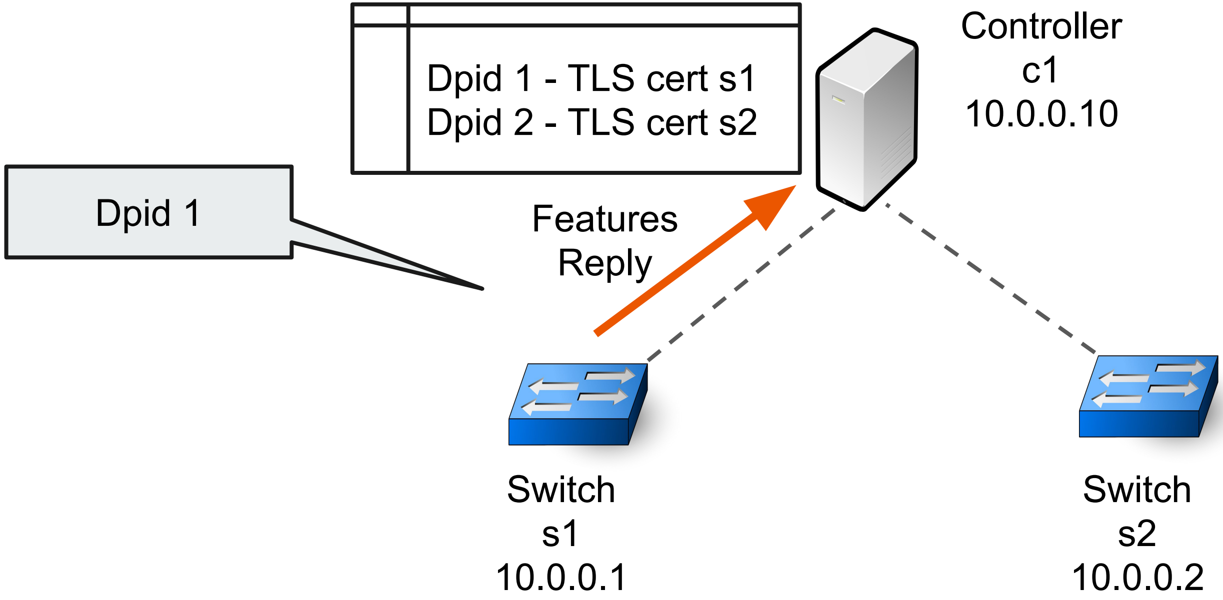
CVE-2018-1000155: Proposed Mitigation

- Unique TLS certificates for switches
- White-list of switch DPIDs at controllers [Gray et al.] and the respective switches' public-key certificate identifier
- A controller mechanism that verifies the DPID announced in the OpenFlow handshake is over the TLS connection with the associated (DPID) certificate
  - ONOS has already patched, see https://github.com/opennetworkinglab/onos/commit/f69e3e34092139600404681798cebeefebcfa6c6
  - Other controllers to follow

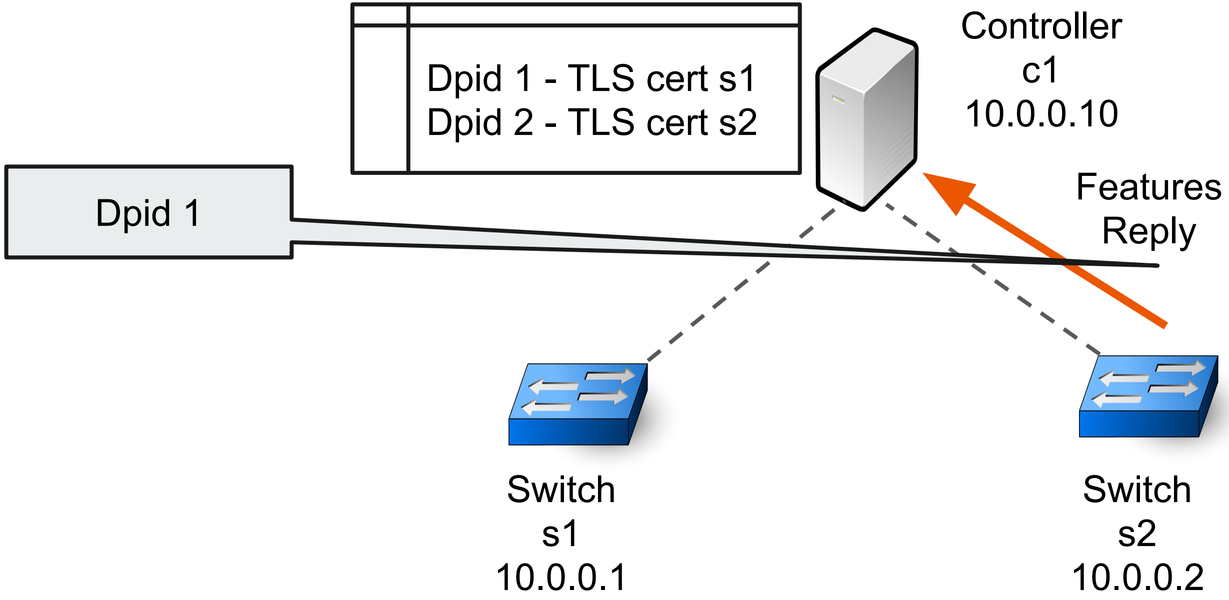
CVE-2018-1000155: Proposed Mitigation



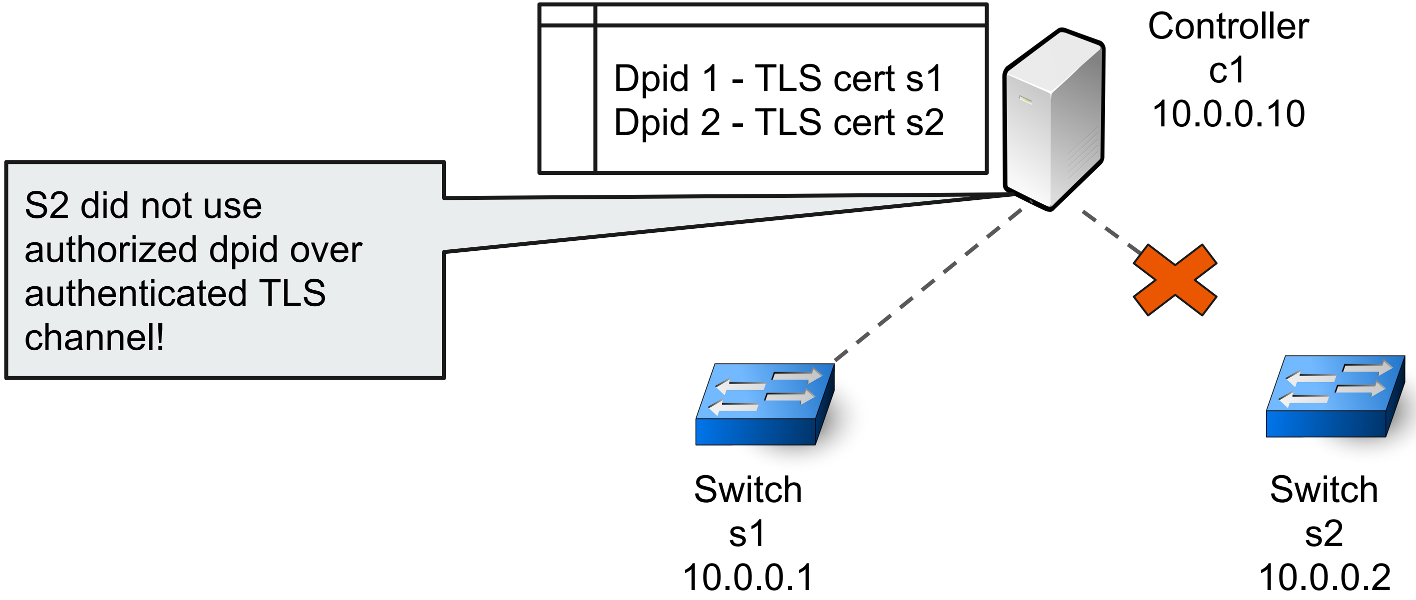
CVE-2018-1000155: Proposed Mitigation



CVE-2018-1000155: Proposed Mitigation



CVE-2018-1000155: Proposed Mitigation



Conclusion

- Introduced a novel covert timing channel in Software-Defined Networks
- A fundamental network security requirement, isolation, can be violated in SDNs using our covert channel
- Our prototype can achieve unidirectional throughput of 20bps with ~90% accuracy
- CVE-2018-1000155 DoS, lack of authentication and authorization, and covert channel in OpenFlow

Contact

Kashyap Thimmaraju

Email: kash@sect.tu-berlin.de

Web: www.fgsect.de/~hashkash

Fingerprint: 5FFC 5589 DC38 F6F5 CEF7 79D8 A10E 670F 9520 75CD

References

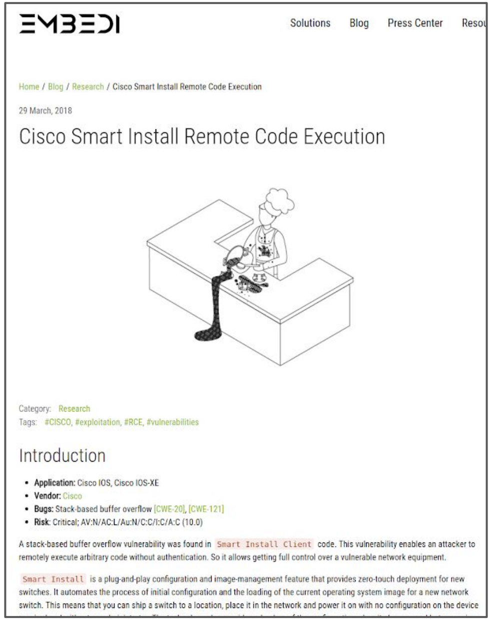
1. [SOSR'18] K. Thimmaraju, B. Shastry, T. Fiebig, F. Hetzelt, J.-P. Seifert, A. Feldmann, S. Schmid," in Proc. ACM Symposium on SDN Research (SOSR), 2018.
2. [EuroSP'17] K. Thimmaraju, L. Schiff, and S. Schmid, "Outsmarting network security with sdn teleportation," in Proc. IEEE European Security & Privacy (S&P), 2017.
3. [Gray et al.] N. Gray, T. Zinner, and P. Tran-Gia, "Enhancing sdn security by device fingerprinting," In Proc. IFIP/IEEE International Symposium on Integrated Network Management (IM), May 2017.
4. [Dover] J. M. Dover, "A denial of service attack against the open floodlight sdn controller," Dover Networks, Tech. Rep., 2013. [Online]. Available: http://dovernetworks.com/wp-content/uploads/ 2013/12/OpenFloodlight-12302013.pdf
5. [Secci et al.] S. Secci, K. Attou, D. C. Phung, S. Scott-Hayward, D. Smyth, S. Vemuri and You Wang, "ONOS Security and Performance Analysis: Report No. 1" ONOS, 2017.
6. [SNBI] https://wiki.opendaylight.org/view/SNBI_Architecture_and_Design
7. [USE] https://wiki.opendaylight.org/images/2/23/Odl-usc-2014_11_20.pdf

Backup

Threats of Switch Id Teleportation

- Stealing private keys
- MITM future traffic
- Fake vpn gateway
- Send control messages as part of a botnet
- Surveillance
- Exfiltration from air-gapped networks with same controller
- Violate network isolation, fundamental requirement.
- Physical isolation via disconnected data planes
- Communication via controller across disconnected data planes
- Why break isolation is bad?
- Break in non-obvious way
    - Fundamental security property broken
    - Physically separated
- Isolation is most basic and required in a network

A More Recent Incident with Cisco



ƎMBƎDI

Solutions Blog Press Center Resou

Home / Blog / Research / Cisco Smart Install Remote Code Execution

29 March, 2018

Cisco Smart Install Remote Code Execution

Category: Research

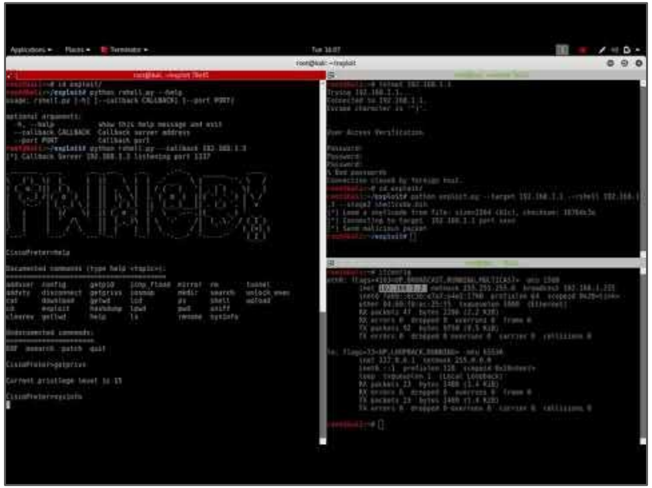
Tags: #CISCO, #exploitation, #RCE, #vulnerabilities

Introduction

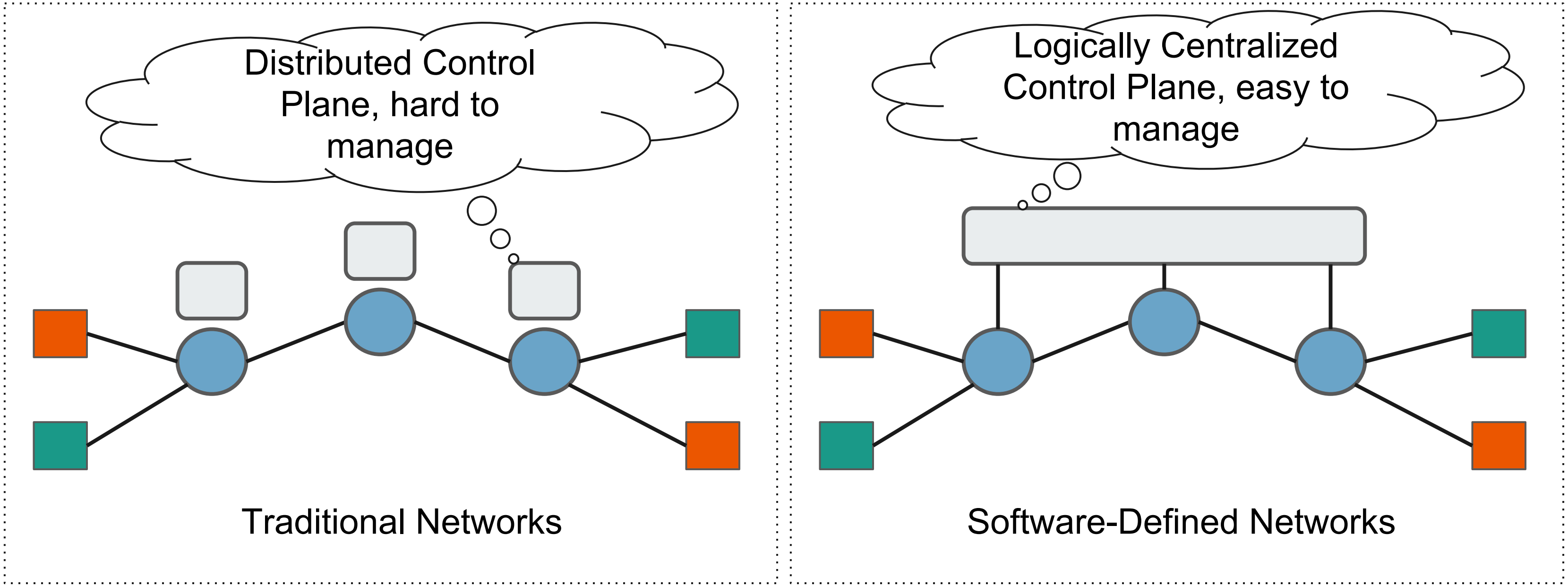
- **Application:** Cisco IOS, Cisco IOS-XE
- **Vendor:** Cisco
- **Bugs:** Stack-based buffer overflow [CWE-20], [CWE-121]
- **Risk:** Critical; AV:N/AC:L/Au:N/C:C/I:C/A:C (10.0)

A stack-based buffer overflow vulnerability was found in `Smart Install Client` code. This vulnerability enables an attacker to remotely execute arbitrary code without authentication. So it allows getting full control over a vulnerable network equipment.

`Smart Install` is a plug-and-play configuration and image-management feature that provides zero-touch deployment for new switches. It automates the process of initial configuration and the loading of the current operating system image for a new network switch. This means that you can ship a switch to a location, place it in the network and power it on with no configuration on the device



Software-Defined Networks (SDN)



Teleportation and OOBF

Teleportation Poses Several Threats

- Bypass security mechanisms
  - Circumvent Firewalls and Intrusion Detection Systems
- Eavesdrop
  - Modify the content of packets in transit

EuroSP'17 focused on Out-of-band Forwarding

Teleportation Poses Several Threats

- Bypass security mechanisms
  - Circumvent Firewalls and Intrusion Detection Systems
- Eavesdrop
  - Modify the content of packets in transit

Networking'18 focuses on Switch Identification

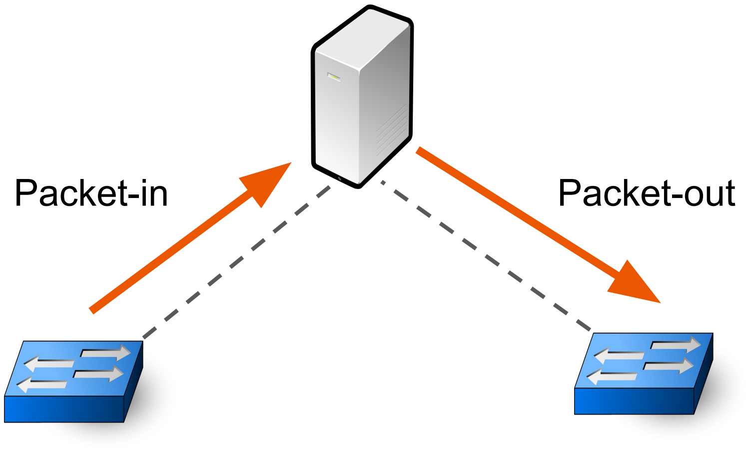
EuroSP'17 focused on Out-of-band Forwarding

- Exfiltration
  - **Violate physical/logical network isolation**
  - **Transmit confidential information, e.g., RSA private keys**
- Attack coordination
  - Send/Receive command and control messages from a Bot master

Out-of-band Forwarding

OpenFlow Messages

- Packet-in
- Packet-out
- (Flow-mods)

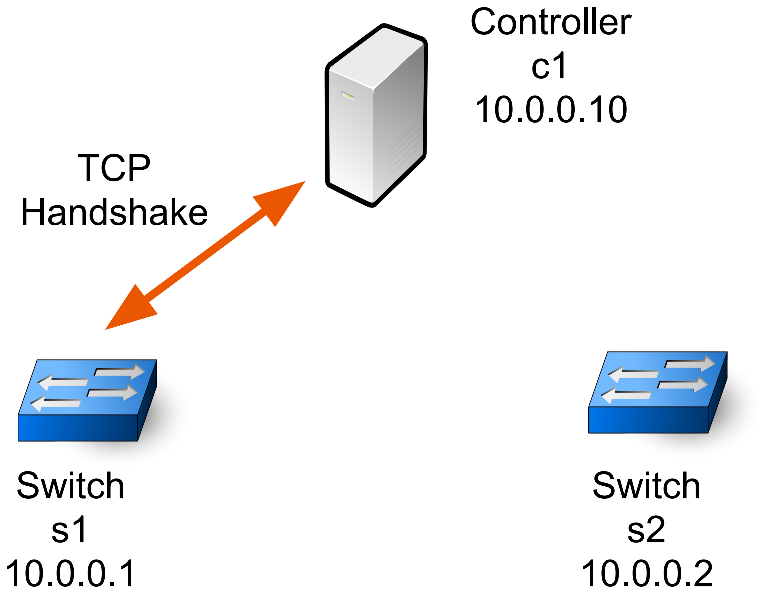


Message Sequence Pattern

Switch Identification Teleportation

OpenFlow Messages

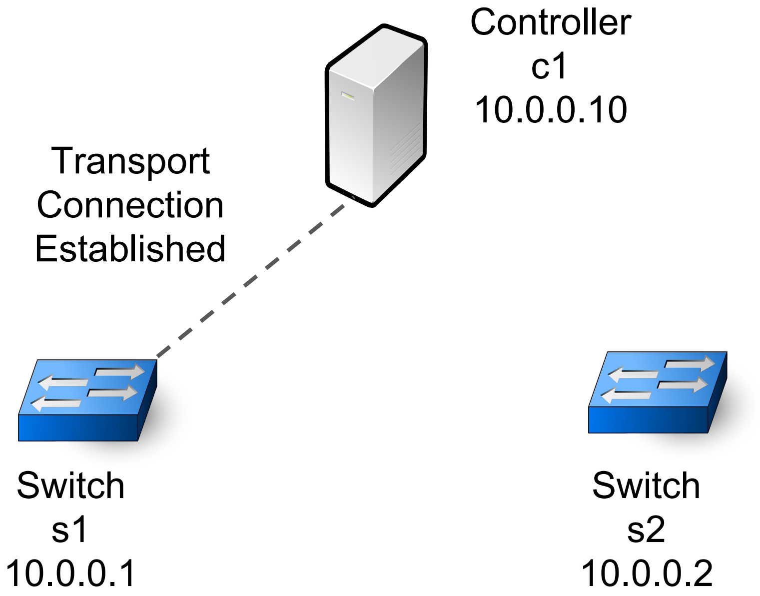
- Hello
- Features Request
- Features Reply



Switch Identification Teleportation

OpenFlow Messages

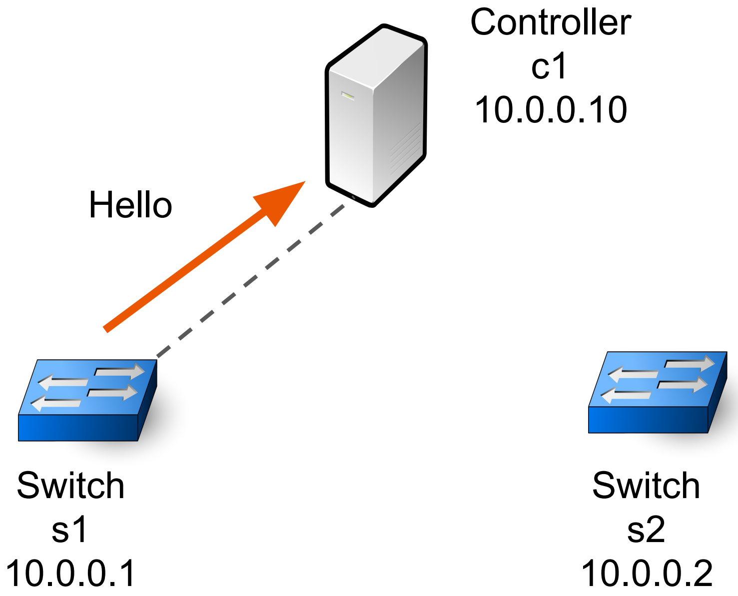
- Hello
- Features Request
- Features Reply



Switch Identification Teleportation

OpenFlow Messages

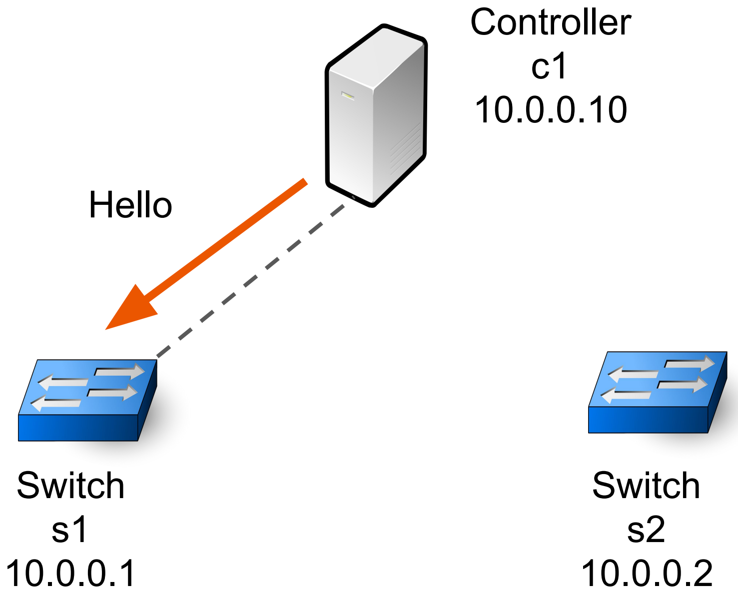
- Hello
- Features Request
- Features Reply



Switch Identification Teleportation

OpenFlow Messages

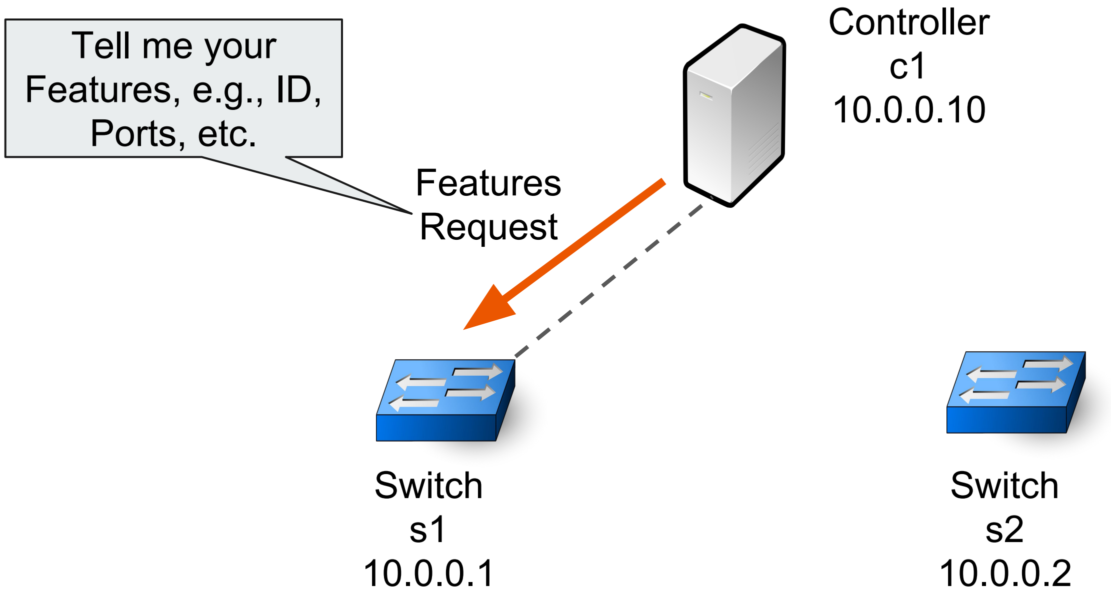
- Hello
- Features Request
- Features Reply



Switch Identification Teleportation

OpenFlow Messages

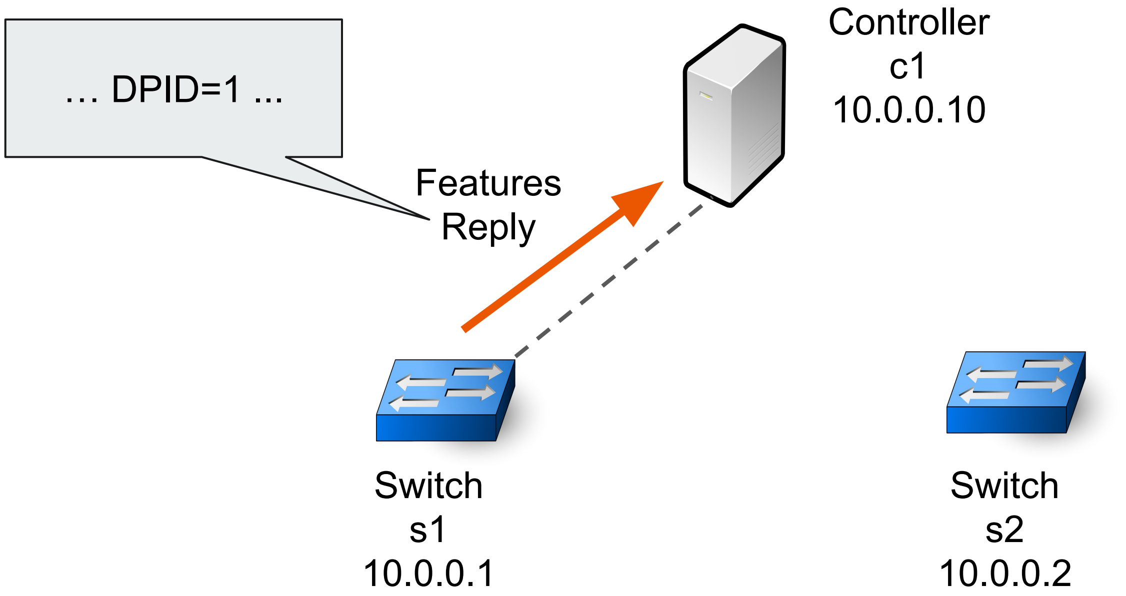
- Hello
- Features Request
- Features Reply



Switch Identification Teleportation

OpenFlow Messages

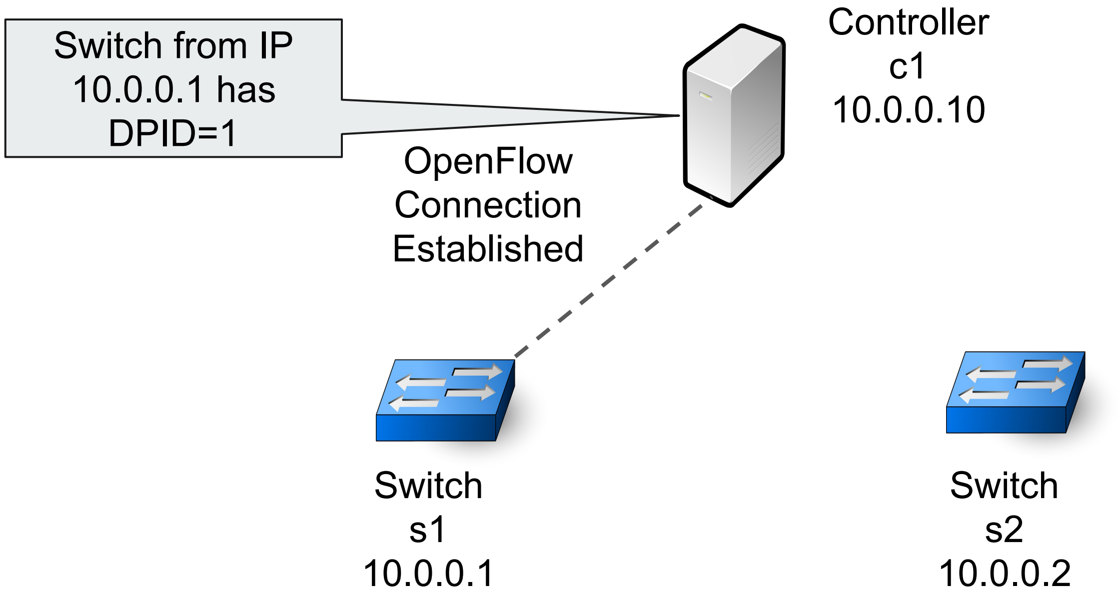
- Hello
- Features Request
- Features Reply



Switch Identification Teleportation

OpenFlow Messages

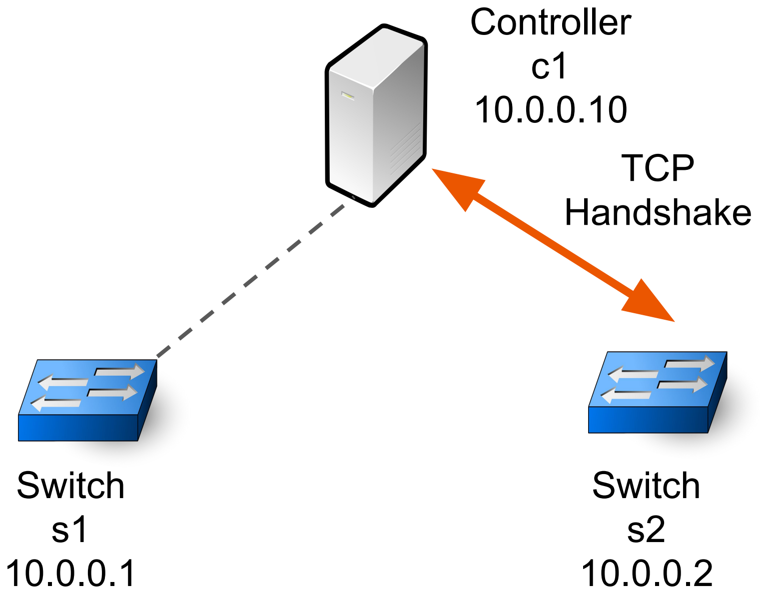
- Hello
- Features Request
- Features Reply



Switch Identification Teleportation

OpenFlow Messages

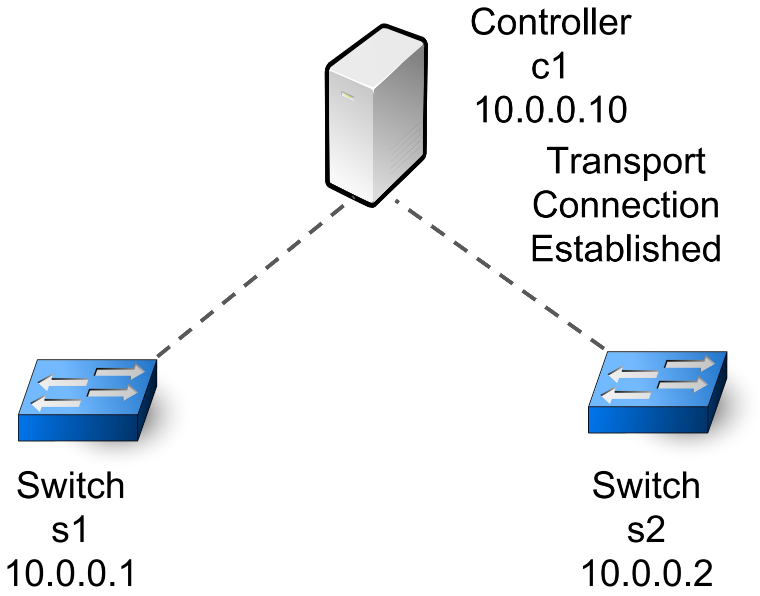
- Hello
- Features Request
- Features Reply



Switch Identification Teleportation

OpenFlow Messages

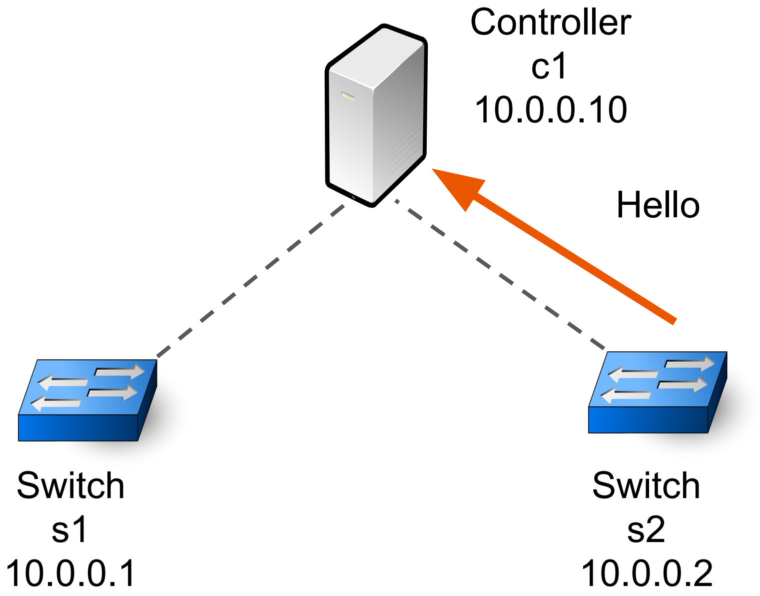
- Hello
- Features Request
- Features Reply



Switch Identification Teleportation

OpenFlow Messages

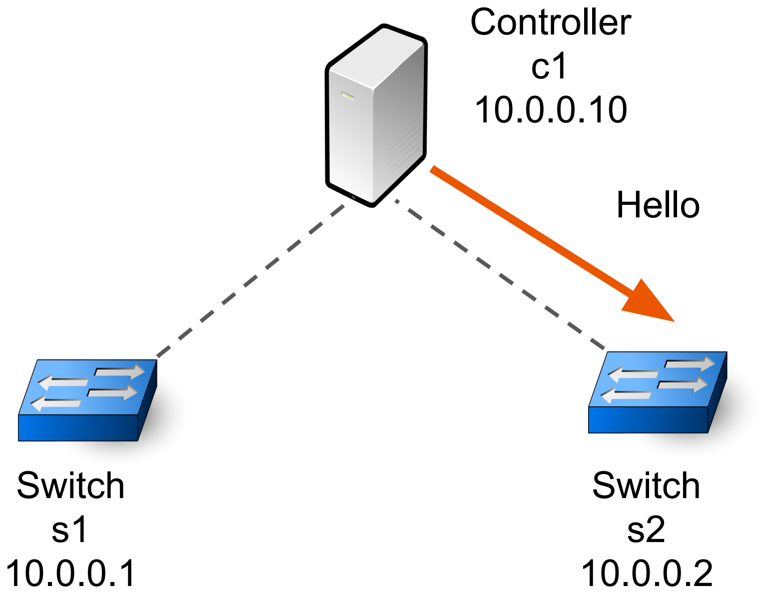
- Hello
- Features Request
- Features Reply



Switch Identification Teleportation

OpenFlow Messages

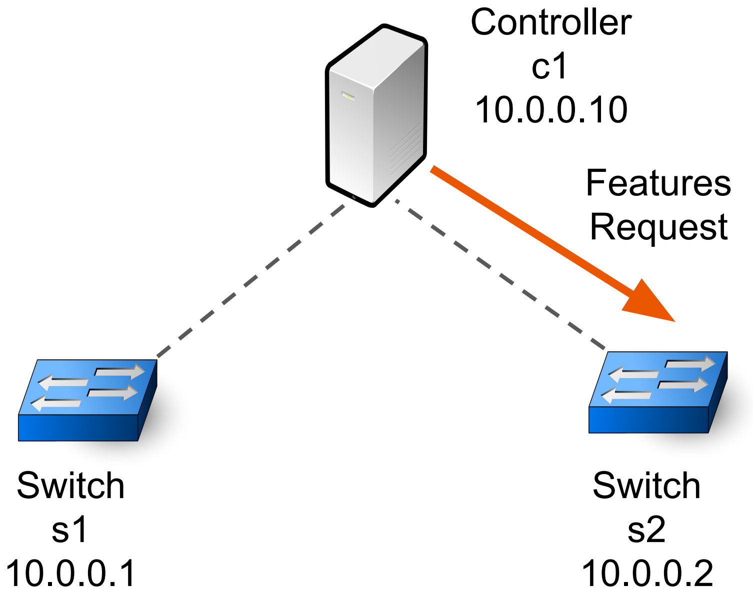
- Hello
- Features Request
- Features Reply



Switch Identification Teleportation

OpenFlow Messages

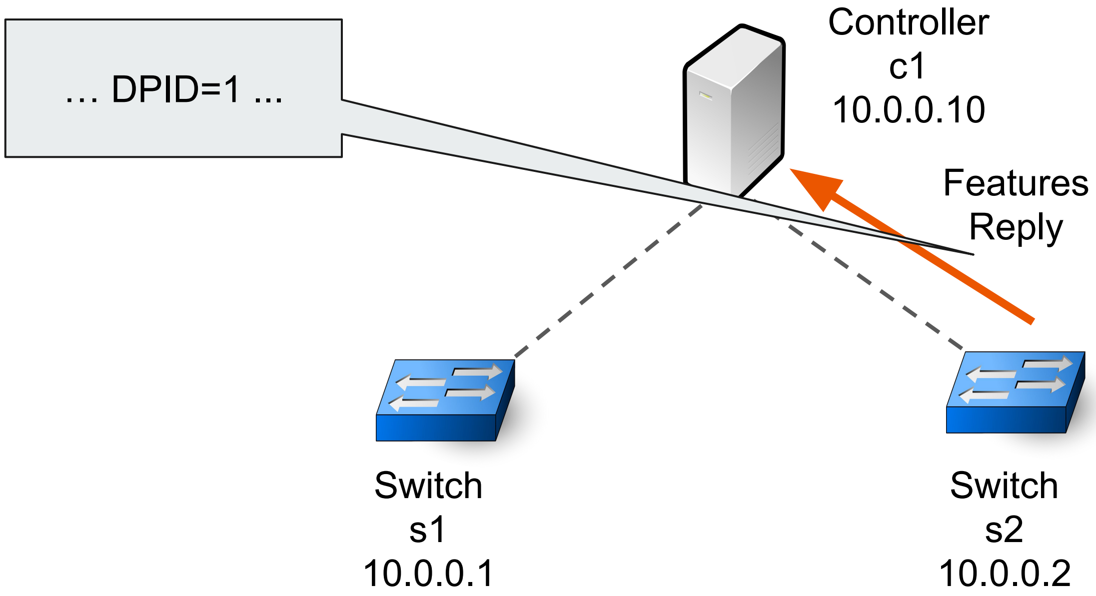
- Hello
- Features Request
- Features Reply



Switch Identification Teleportation

OpenFlow Messages

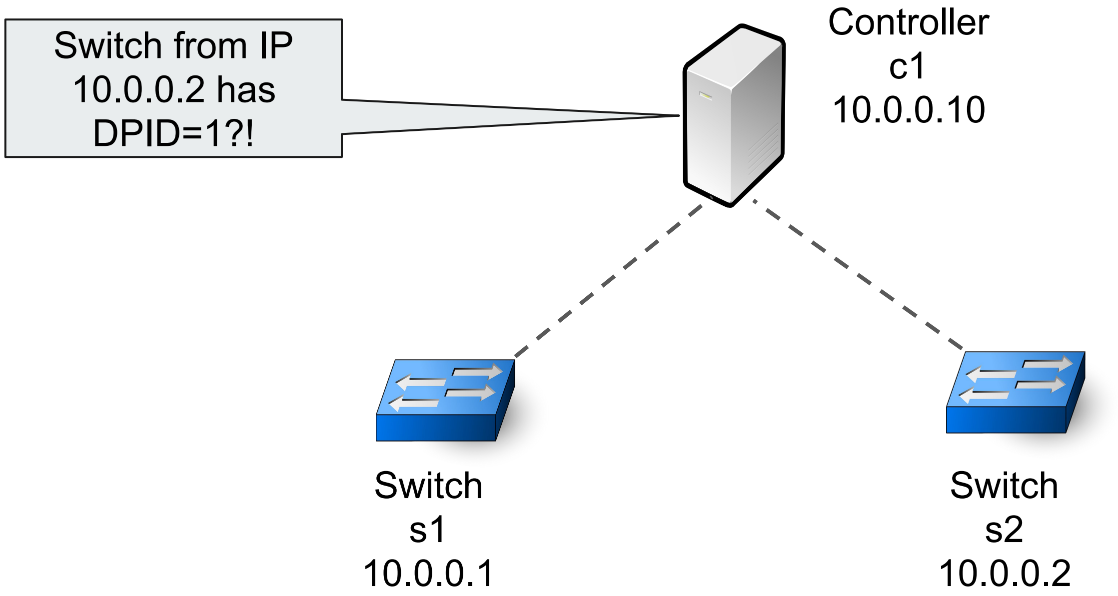
- Hello
- Features Request
- Features Reply



Switch Identification Teleportation

OpenFlow Messages

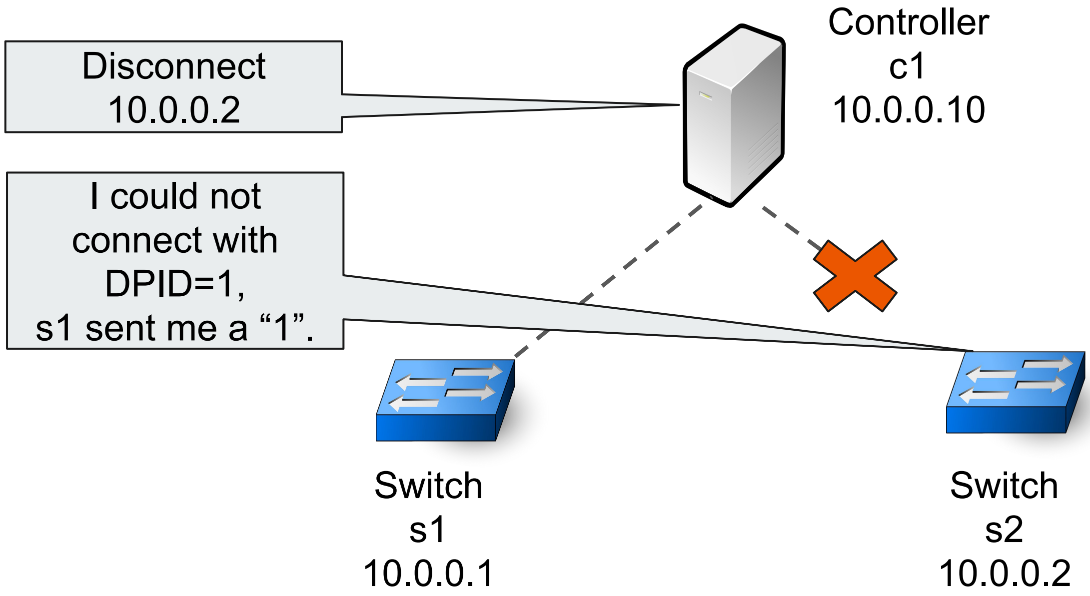
- Hello
- Features Request
- Features Reply



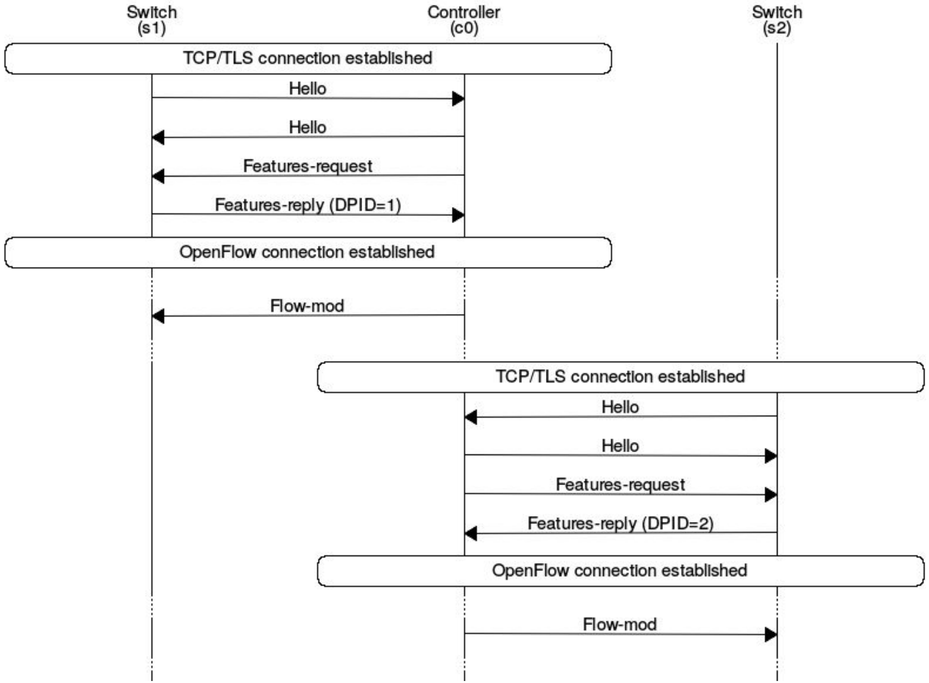
Switch Identification Teleportation

OpenFlow Messages

- Hello
- Features Request
- Features Reply

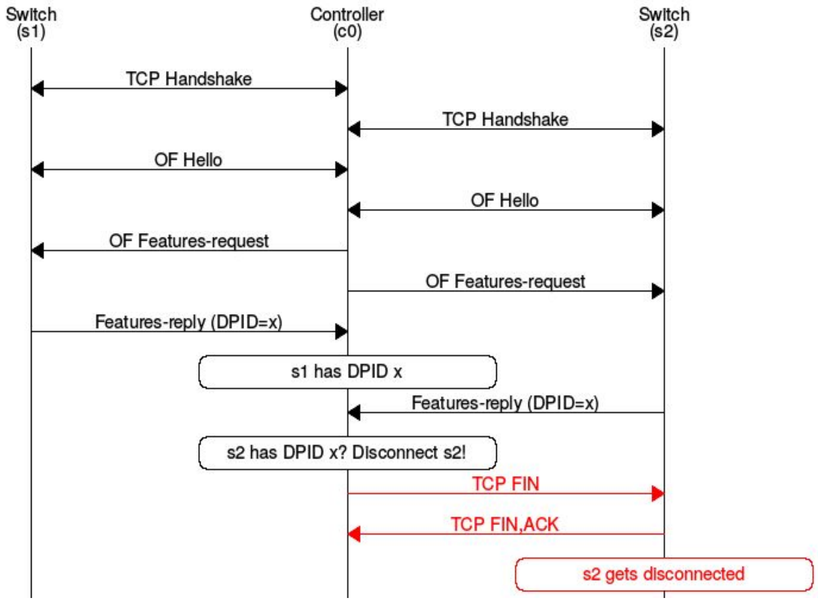


OpenFlow Handshake



Switch Identification Teleportation

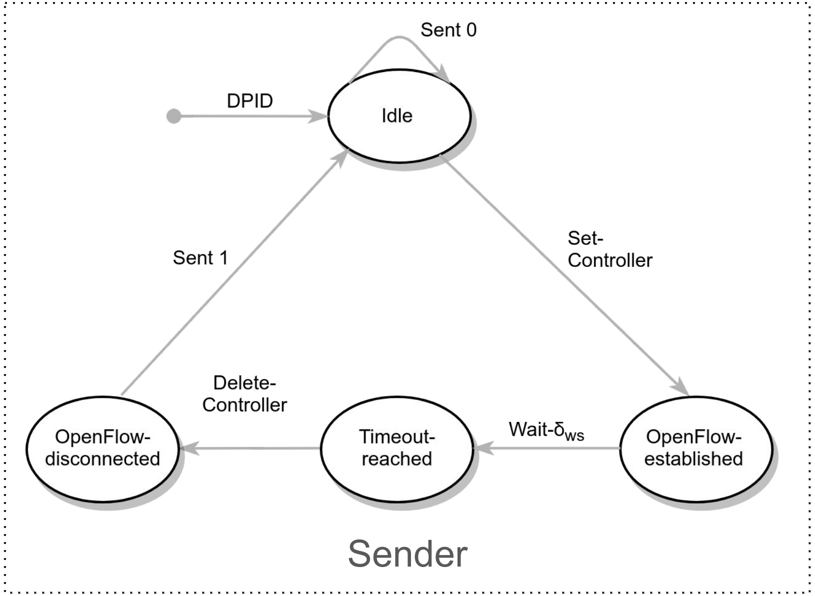
With ONOS

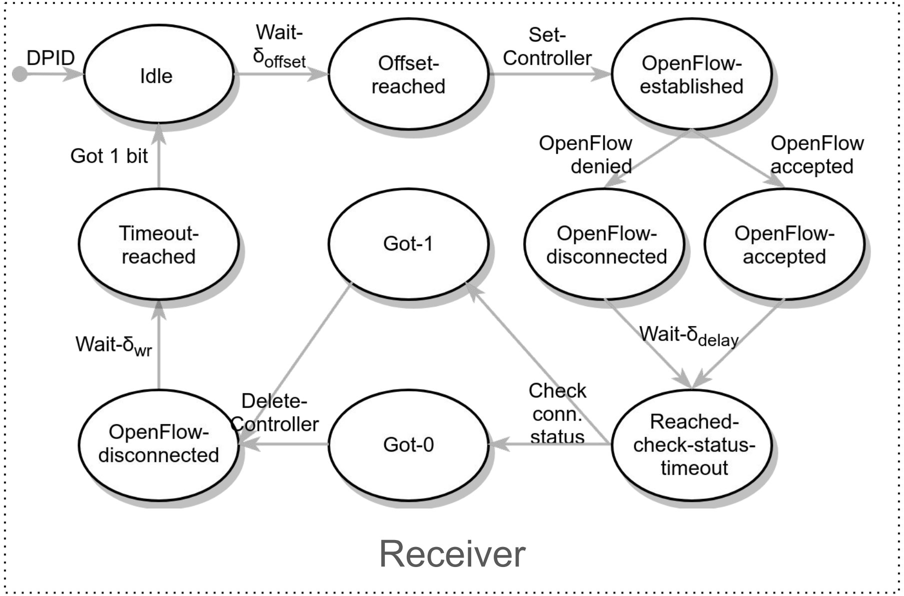


OpenFlow Handshake

State Transition Model

State Transition Model





Transition Delays

Transition Delays

1. δ_s : The time the sender takes to send a binary bit value
2. δ_r : The time the receiver takes to receive a binary bit value
3. δ_{sc} : The time to transition from the Idle state to the OpenFlow-established state
4. δ_{dc} : The time to move from the OpenFlow-established state to OpenFlow-disconnected state
5. δ_{offset}: A timeout value the receiver waits before it sets the controller
6. $\delta_{of\text{-}deny}$: The time to move from OpenFlow-established to OpenFlow-disconnected when the connection is denied
7. δ_{delay} : A timeout value the receiver waits before it checks the OpenFlow connection status
8. $\delta_{chk\text{-}conn}$: The time the receiver takes to determine a 0 or 1 by checking the OpenFlow connection status
9. $\delta_{ws} = \Delta - \delta_s$: A timeout value the sender waits before moving from the OpenFlow-established state to OpenFlow-disconnected
10. $\delta_{wr} = \Delta - \delta_r$: A timeout value the receiver waits before moving from the OpenFlow-disconnected state to the Idle state

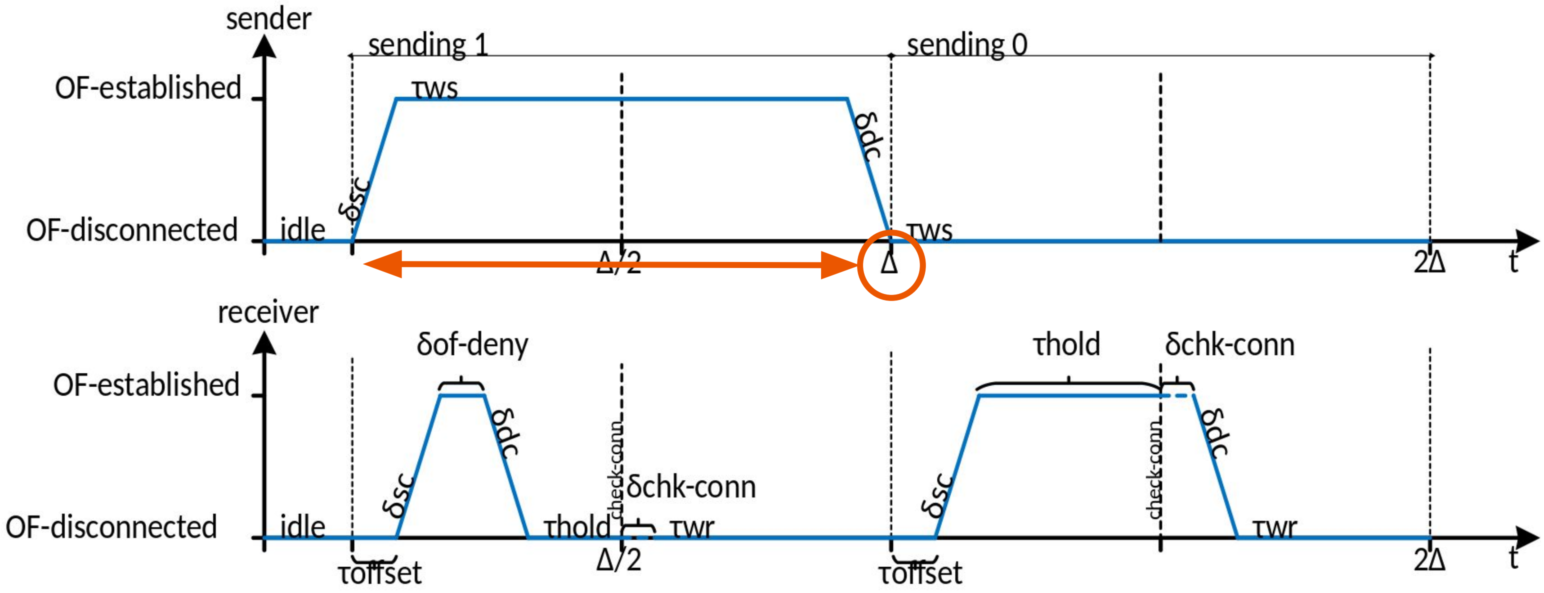
Boundary Conditions

Experiments

Measured the accuracy using Levenshtein distance

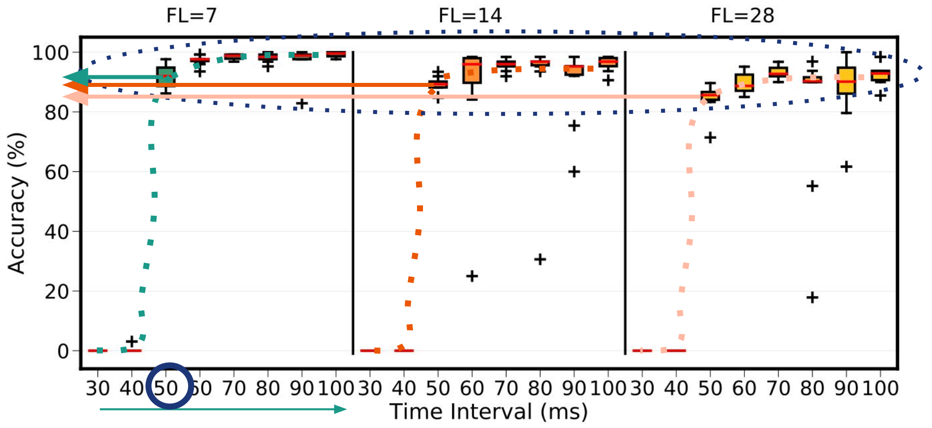
1. Effect of timing interval (Δ)
2. Effect of frame length (FL)
3. Effect of delay in conn. Status (δ_{delay})
4. Effect of load on the controller
5. Effect of message length (M)

Timing Diagram



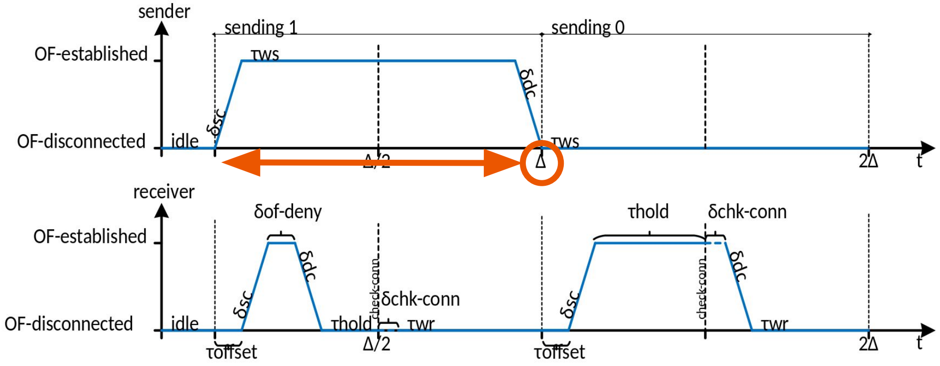
Effect of Timing Interval (Δ) and Frame Length (FL)

No load, M=64bytes, δ_{offset}=5ms and check the conn. status at Δ/2



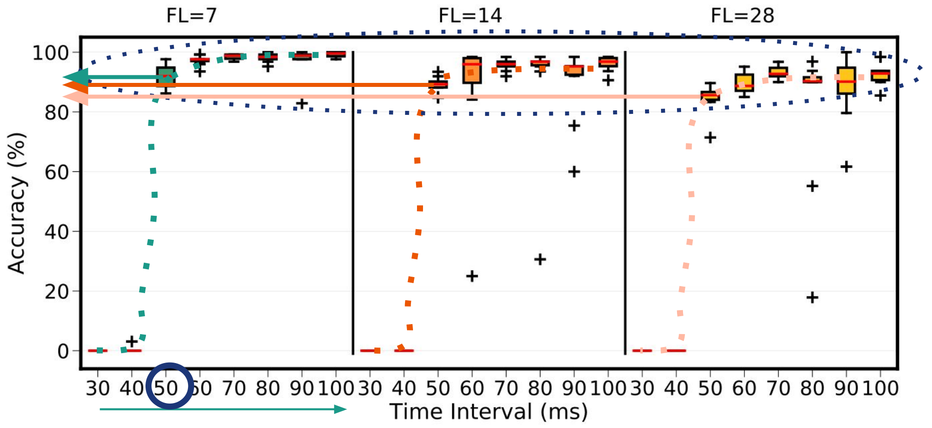
Effect of Timing Interval (Δ) and Frame Length (FL)

No load, M=64bytes, δ_{offset}=5ms and check the conn. status at Δ/2

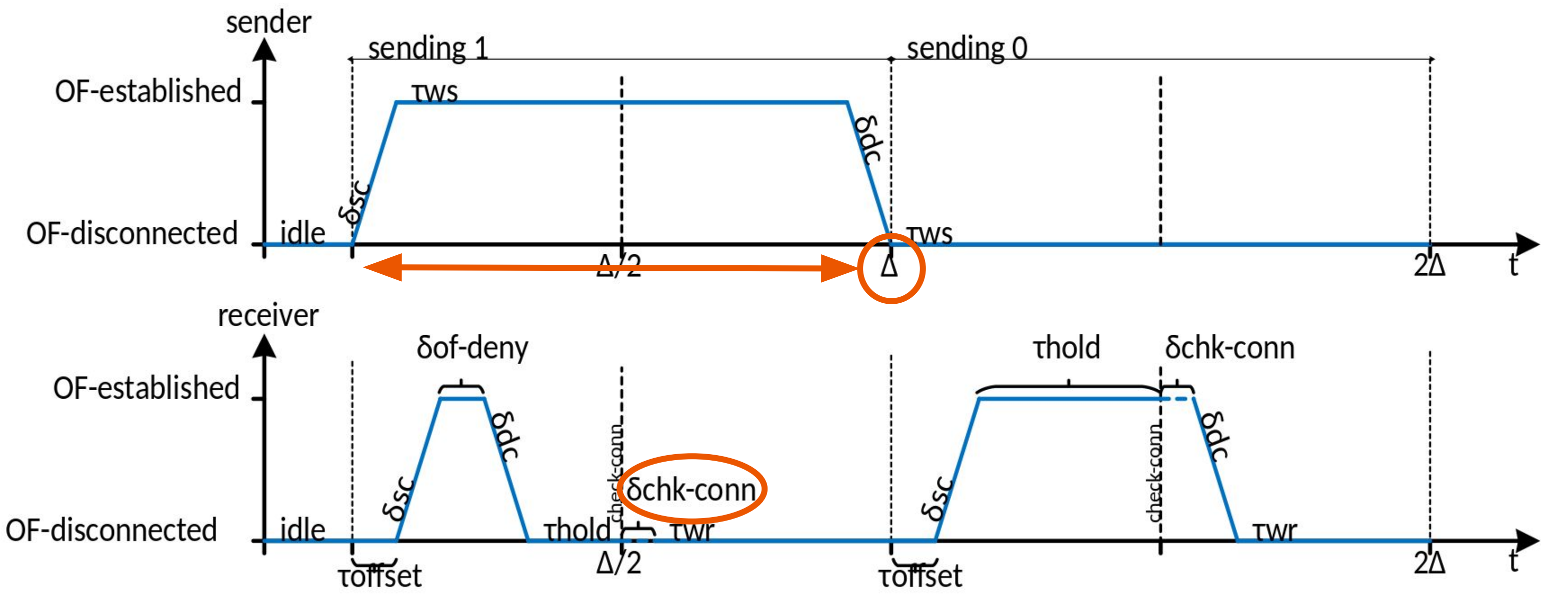


Effect of Timing Interval (Δ) and Frame Length (FL)

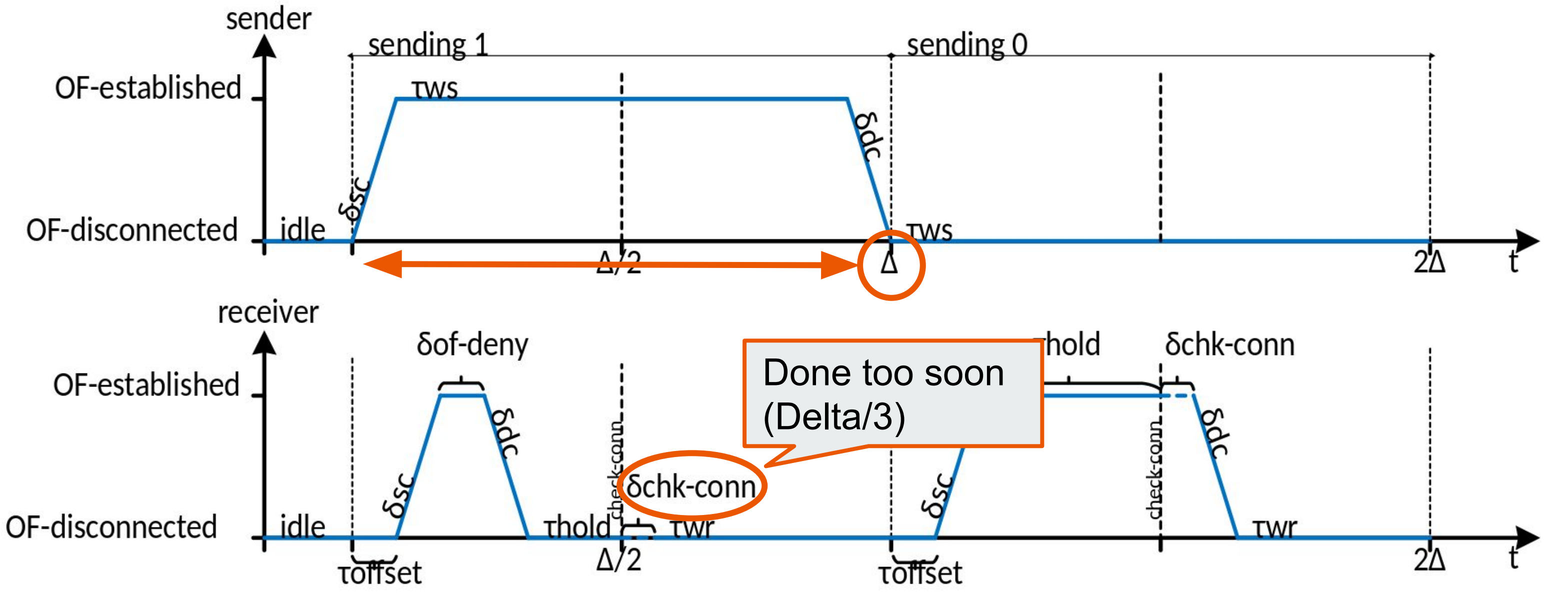
No load, M=64bytes, δ_{offset}=5ms and check the conn. status at Δ/2



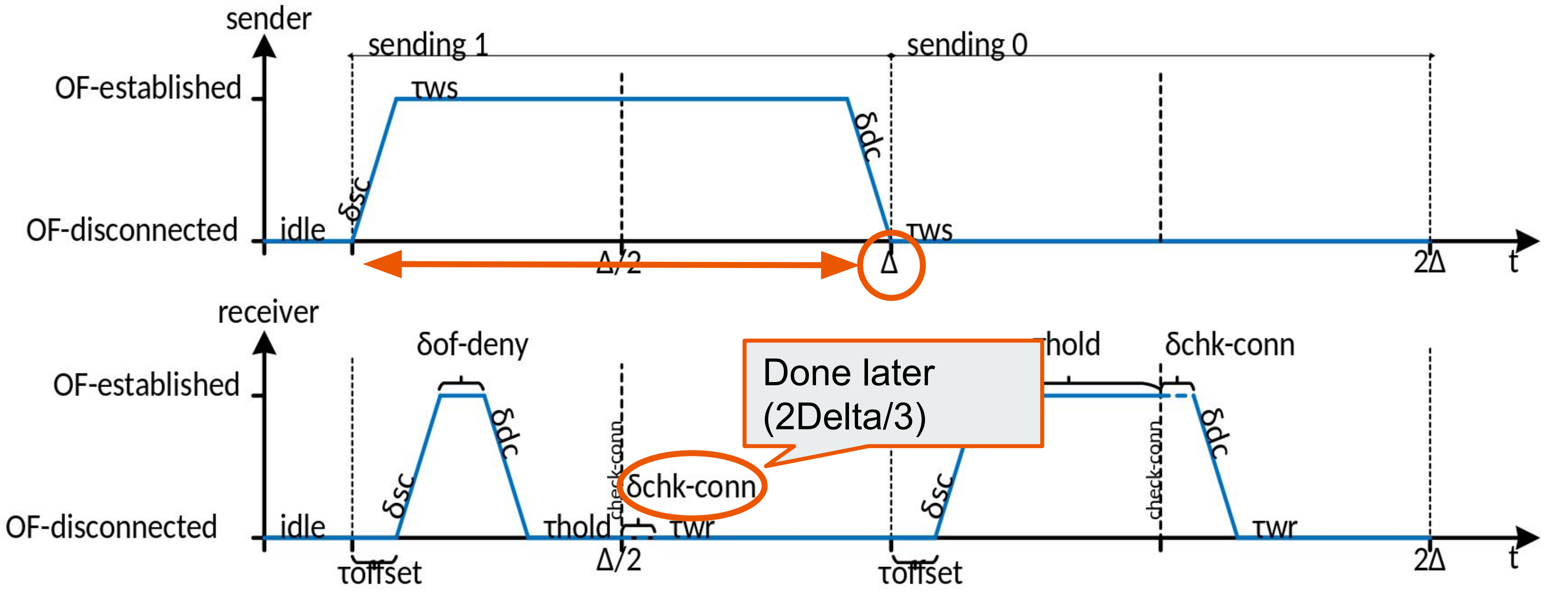
Timing Diagram



Timing Diagram

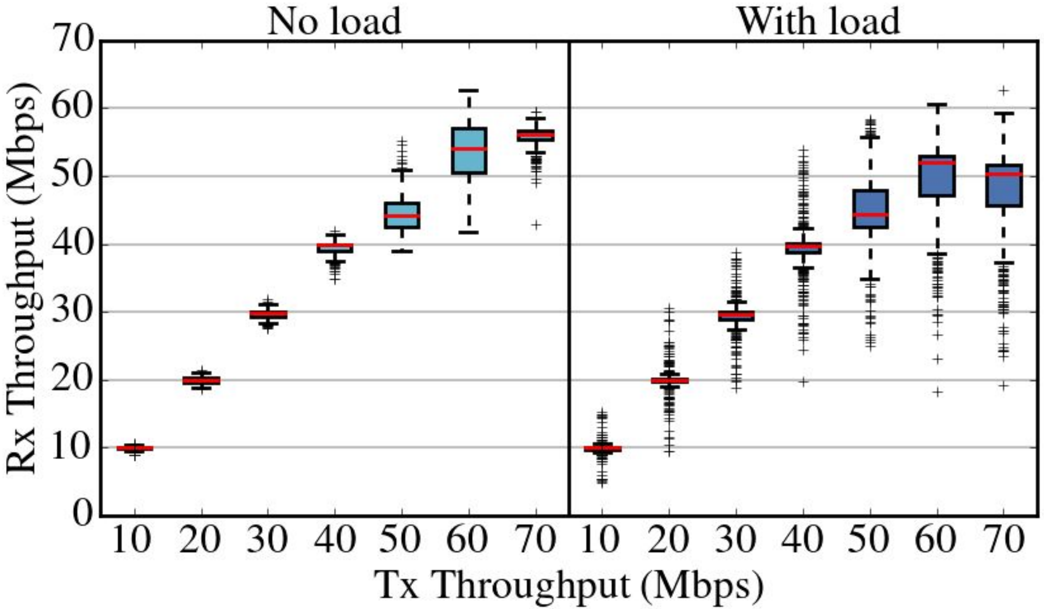


Timing Diagram



OOBF Throughput

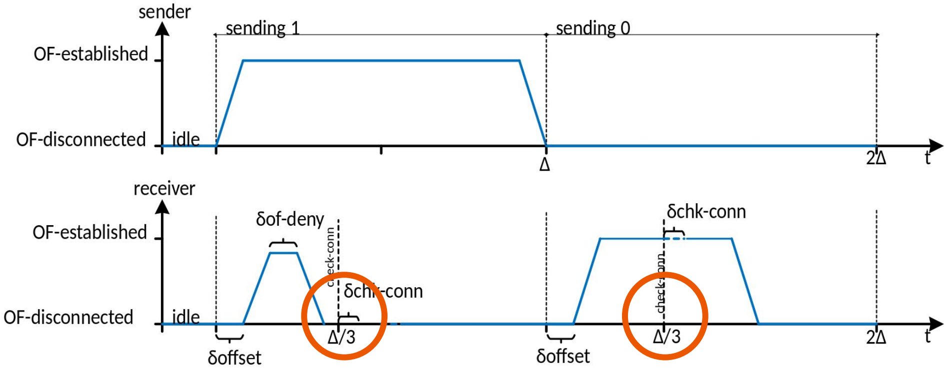
Out-of-band Forwarding Throughput



Accuracy and Error Analysis

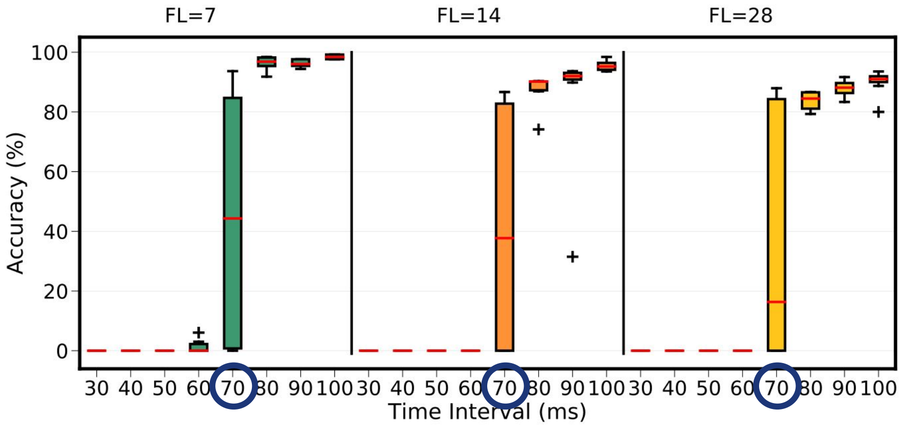
Effect of Delay (δ_{delay}) to Check Conn. Status

No load, M=64bytes, δ_{offset}=5ms and check the conn. status at Δ/3

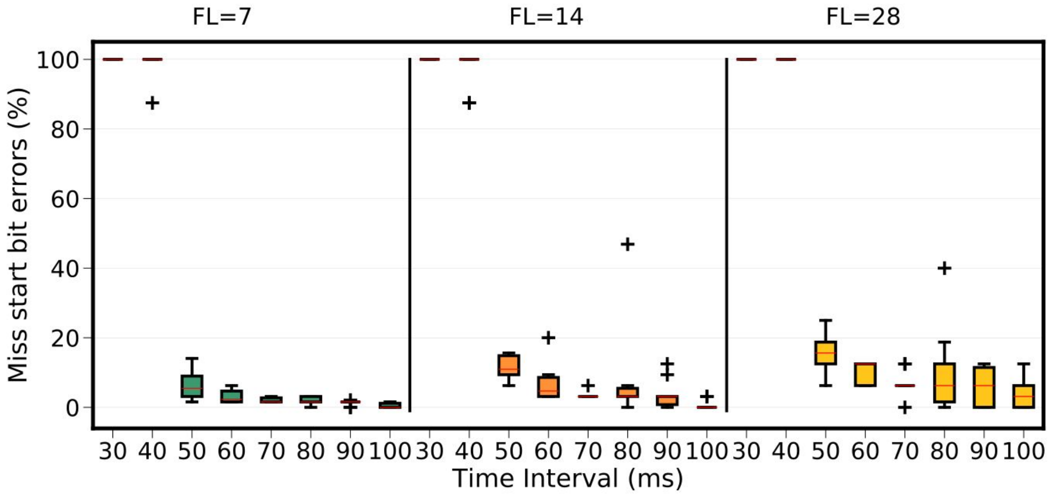


Effect of Delay (δ_{delay}) to Check Conn. Status

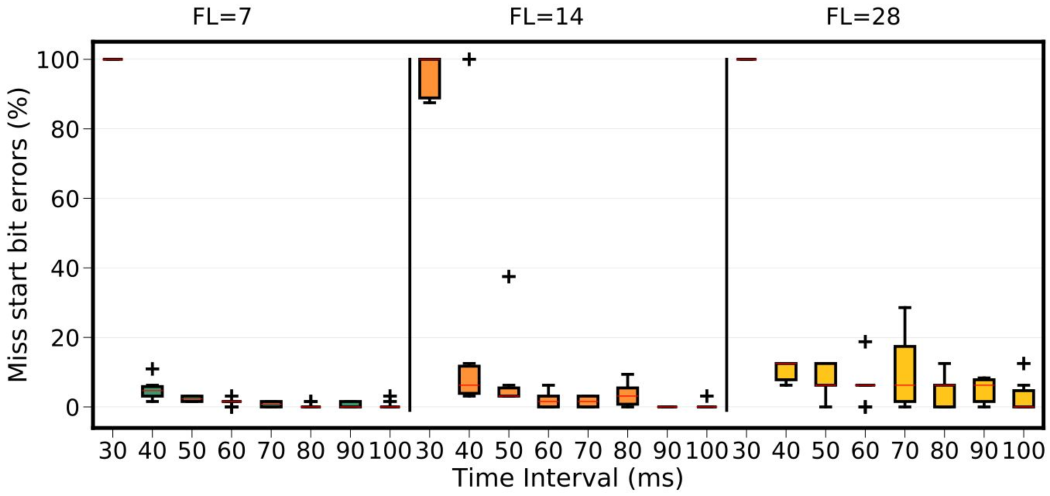
No load, M=64bytes, δ_{offset}=5ms and check the conn. status at $\Delta/3$



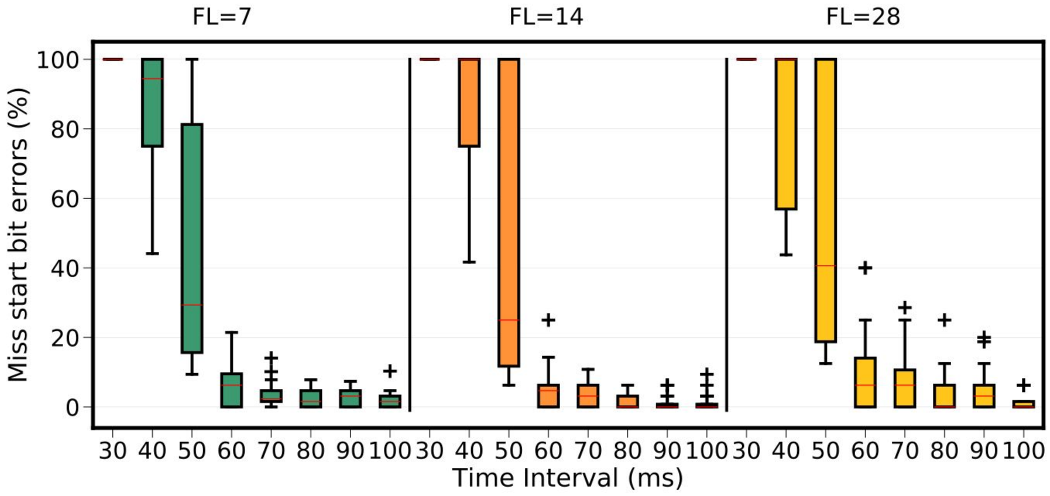
Miss Start Bit Error: noLoad, 2.0d



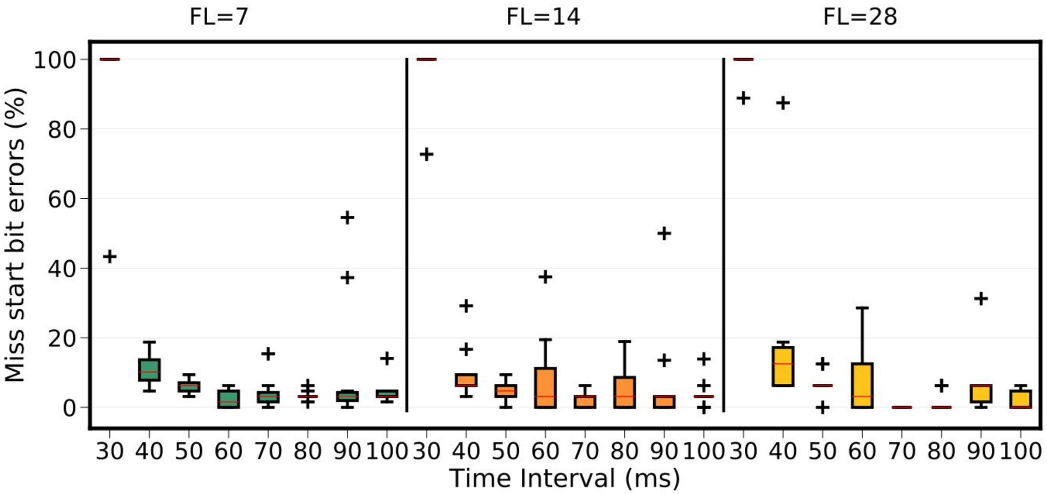
Miss Start Bit Error: noLoad, 2/3d



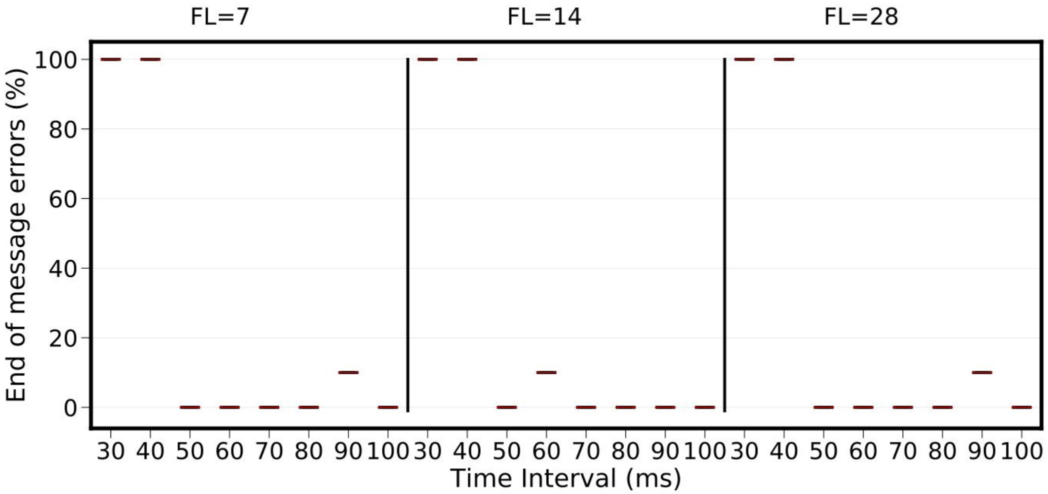
Miss Start Bit Error: withLoad, 2.0d



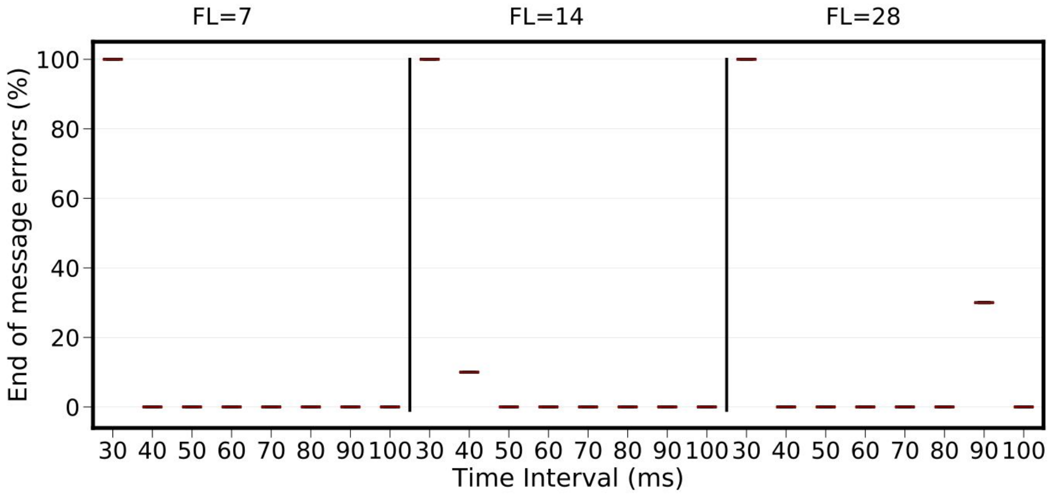
Miss Start Bit Error: withLoad, 2/3d



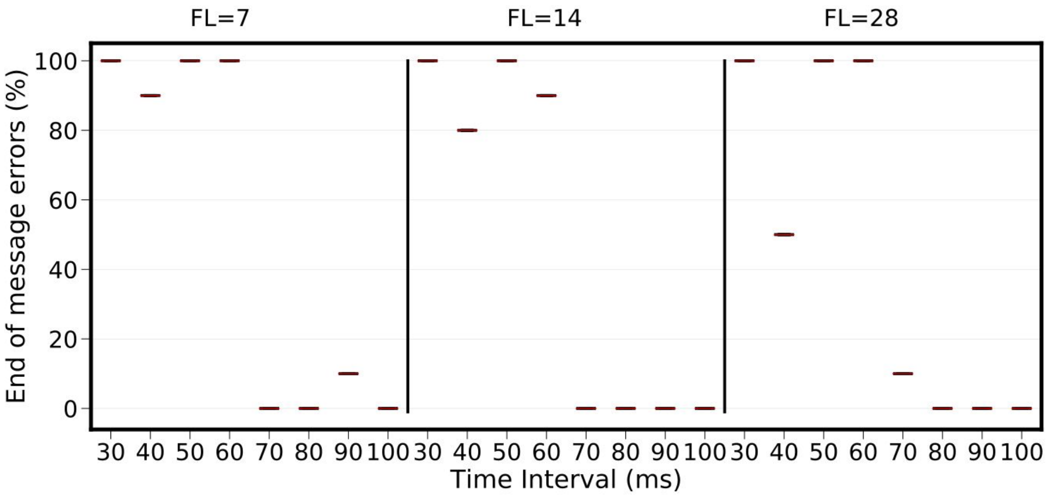
End of Message Error: noLoad, 2.0d



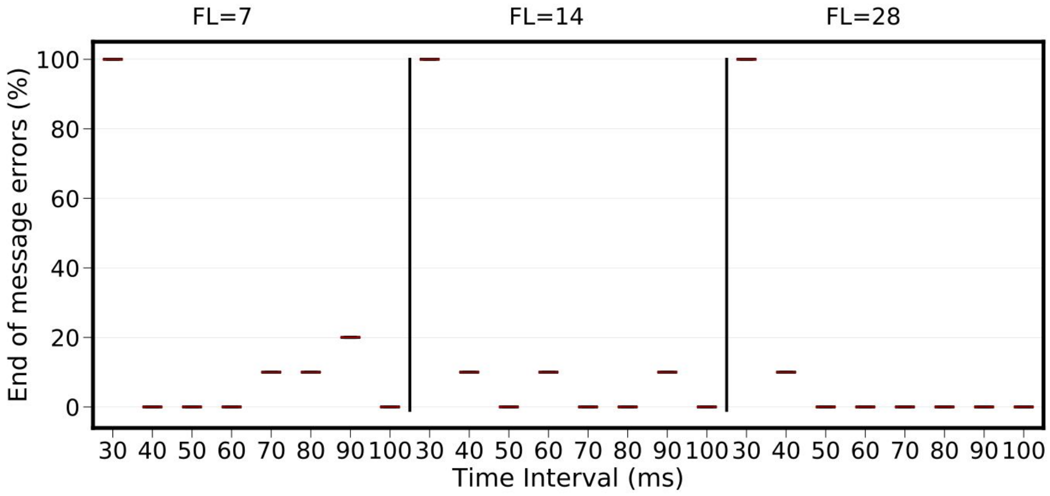
End of Message Error: noLoad, 2/3d



End of Message Error: withLoad, 2.0d



End of Message Error: withLoad, 2/3d

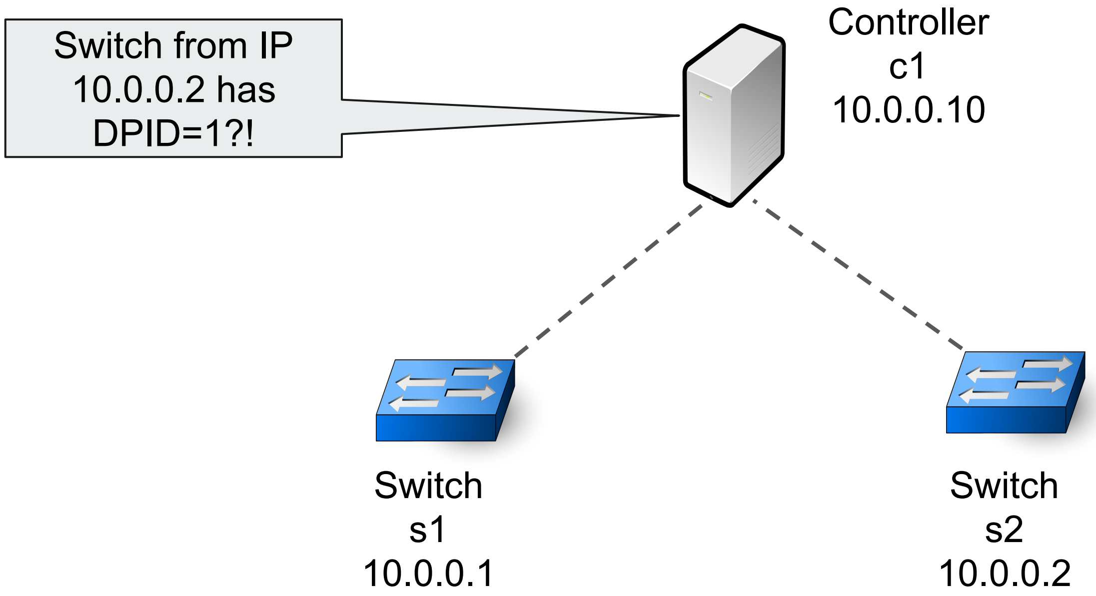


Why TLS is Insufficient

Switch Identification Teleportation

OpenFlow Messages

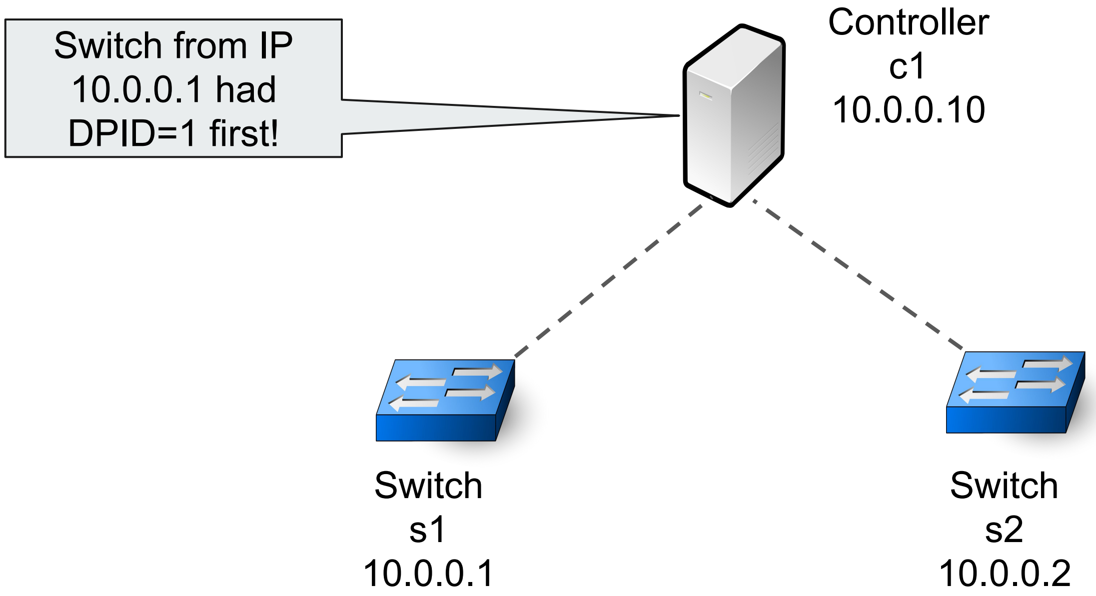
- Hello
- Features Request
- Features Reply



Switch Identification Teleportation

OpenFlow Messages

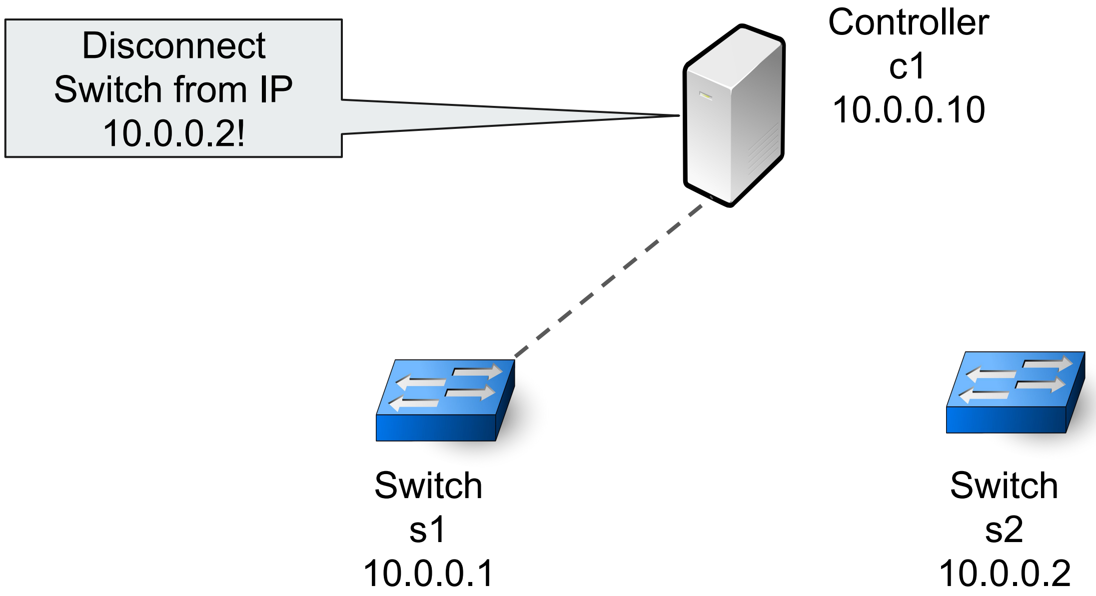
- Hello
- Features Request
- Features Reply



Switch Identification Teleportation

OpenFlow Messages

- Hello
- Features Request
- Features Reply



Switch Identification Teleportation

OpenFlow Messages

- Hello
- Features Request
- Features Reply