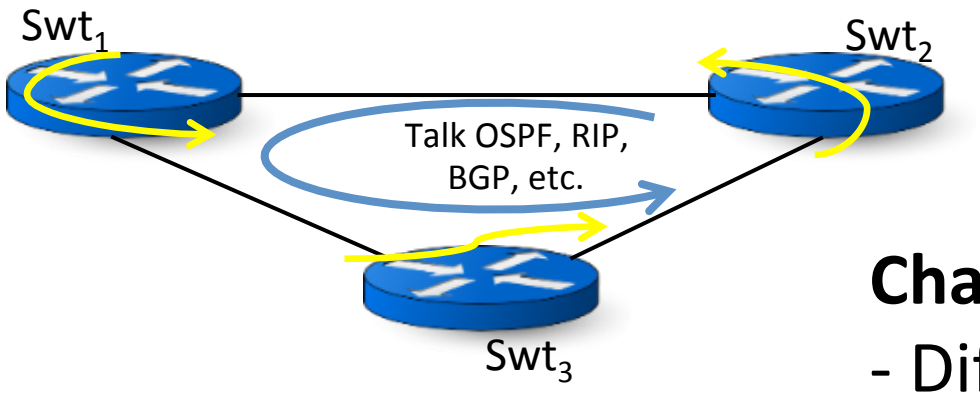
Abstractions for Model Checking SDN Controllers

Divjyot Sethi, Srinivas Narayana,
Prof. Sharad Malik
Princeton University

Traditional Networking



- Forwarding data plane
  - Mapping used for forwarding packets.
- Distributed control plane
  - Logic used to update the mapping.

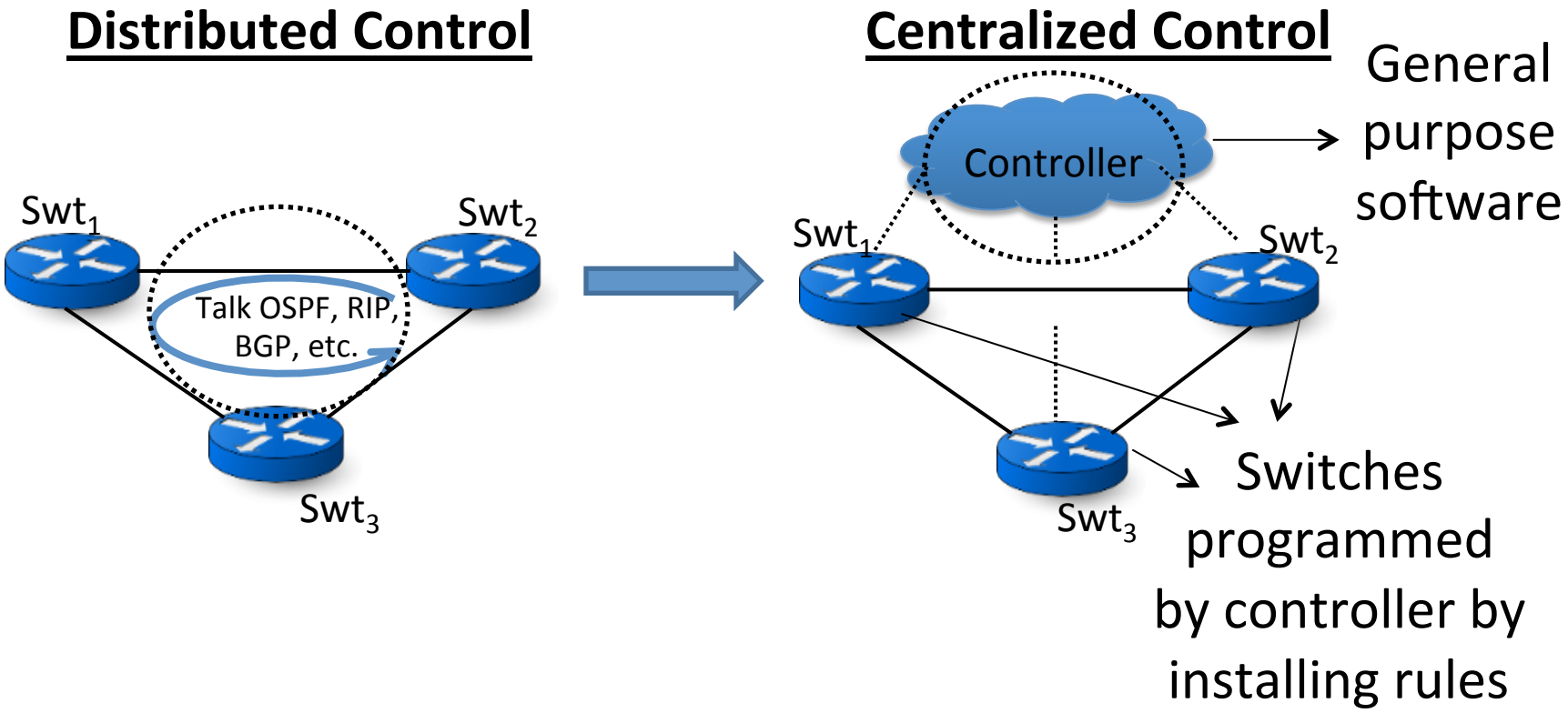
Challenges:

- Difficult to get right.

- Inflexible for novel ideas.

- No clean abstractions for implementing control.

A Fundamental Shift in Network Design



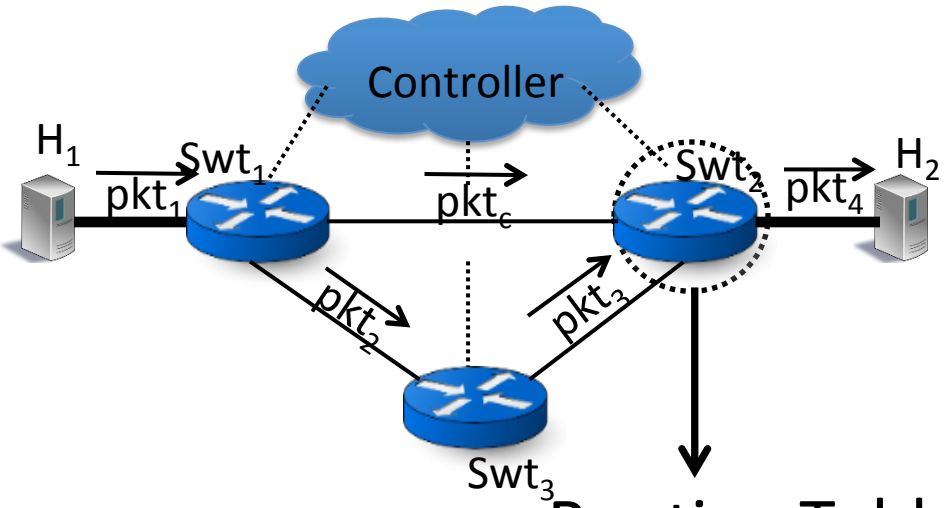
Centralized control simplifies design and innovation
However, an Achilles heel for correctness.

Problem: Bugs in Centralized Control?

- Security leaks: packet sent to an untrusted host.

- Network loops: packet looping around in network.
  - Link overload and data center outage.
    - Downtime cost: ~$1 million per outage! (www.informationweek.com)
    - AWS service commitment: Amazon EC2 and Amazon availability at least 99.95%

Challenges in Verification



Routing Table

outPort(inPkt) =

$Port_1$: inPkt.dst = H_1
$Port_2$: inPkt.dst = H_3
$Port_3$: inPkt.dst = H_k
⋮
$Port_p$: inPkt.dst = H_r
$Port_q$: inPkt.dst = H_a

- Large number of packets alive in network.
  - Large buffer state.
- Large number of rules installed in switches.
  - Large network state.
- Large topology size.

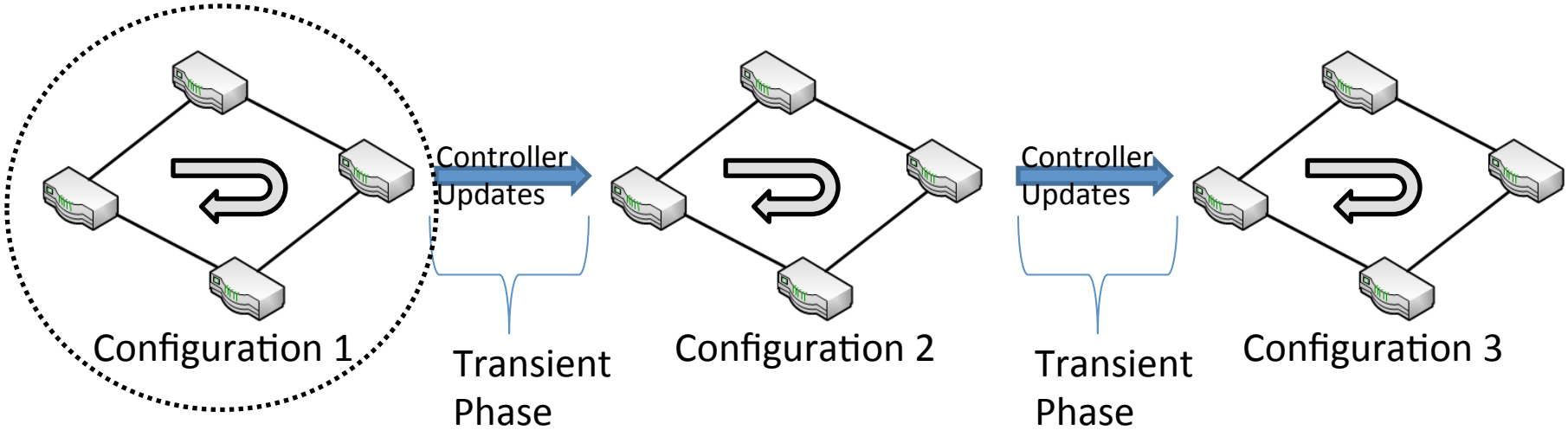
Overview

- Existing approaches and problem statement
- Abstraction on Stateful firewall
- Experimental case studies
  - Stateful firewall
  - Learning switch
- Conclusions

Overview

- **Existing approaches and problem statement**
- Abstraction on Stateful firewall
- Experimental case studies
  - Stateful firewall
  - Learning switch
- Conclusions

Verifying Software Defined Networks: Existing Approaches

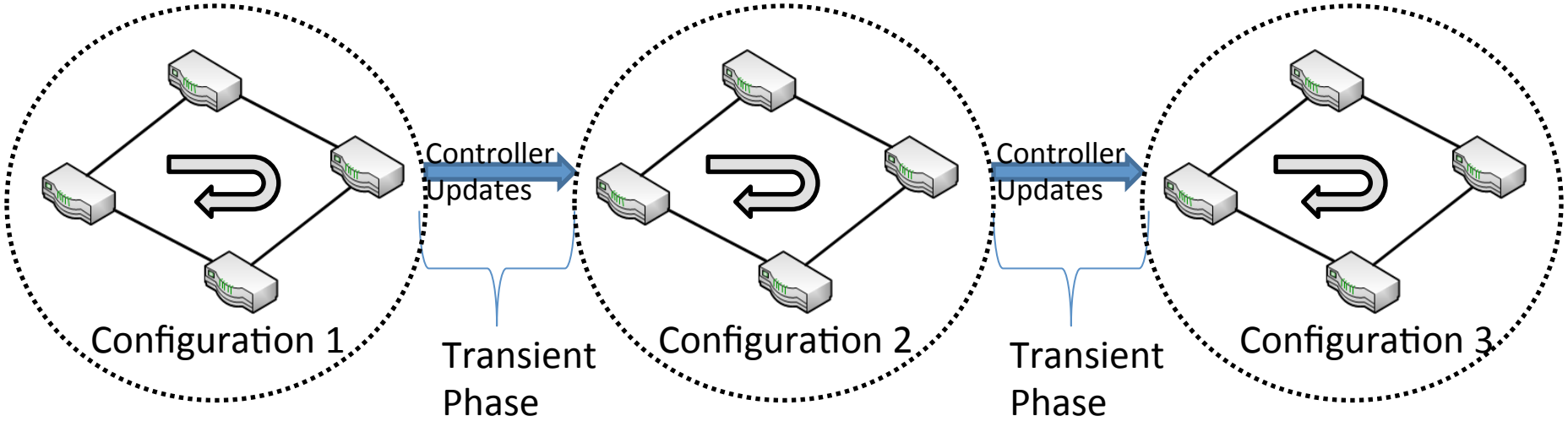


Category 1: Verify just one configuration

- Symbolic simulation[Kazemian et al. NSDI'12]
- Reduction to SAT [S. Zhang et al. ATVA'12, H. Mai SIGCOMM' 11]
- Model Checking [E. Al-Shaer SafeConfig'10]

Problem: verifies just one configuration!

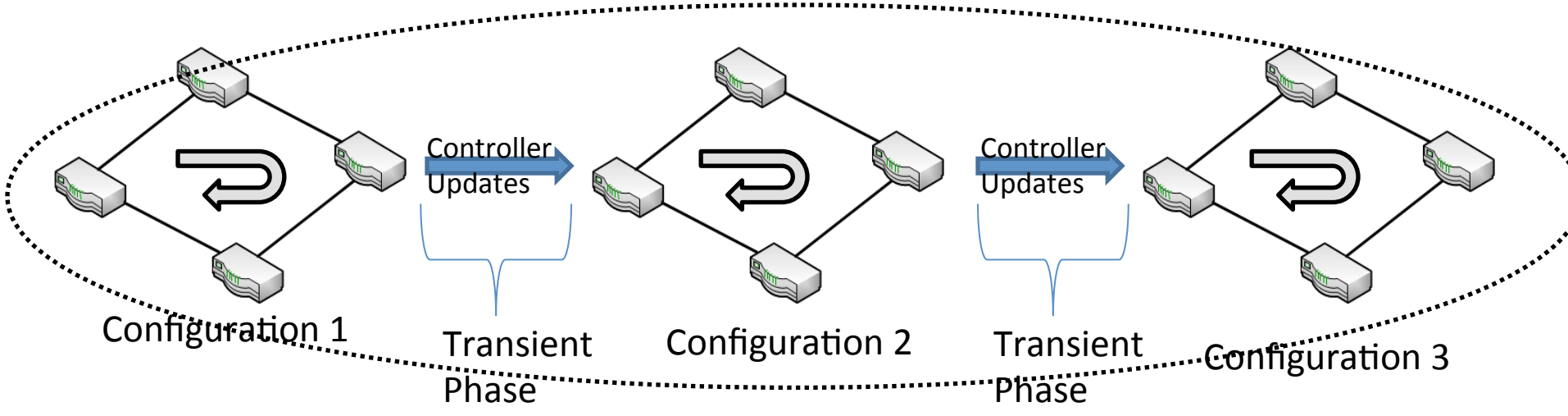
Verifying Software Defined Networks: Existing Approaches



Category 2: Incremental verification, i.e., verify all configurations.
[Kazemian et al. NSDI'13, A. Khurshid et al. NSDI'12]

Problem: property may be violated in transient phase!

Verifying Software Defined Networks: Existing Approaches



Network state evolves from configuration (switch rules) to configuration as controller updates the rules during transient phase.

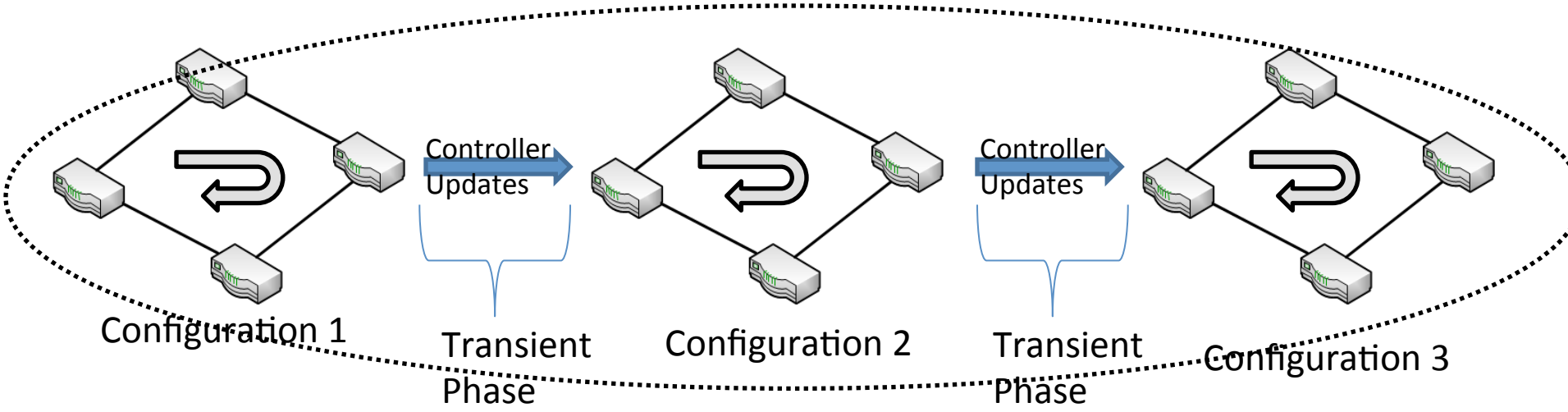
Category 3: Full formal verification of Controller

- NICE (M. Canini NSDI'12), FlowLog (T. Nelson HotSDN'13)

Problem: handle only a bounded number of packets!

- Runtime grows exponentially with increasing packets.
- Can't guarantee properties like security as checked for small number of packets.

Focus of this Work



Network state evolves from configuration (switch rules) to configuration as controller updates the rules during transient phase.

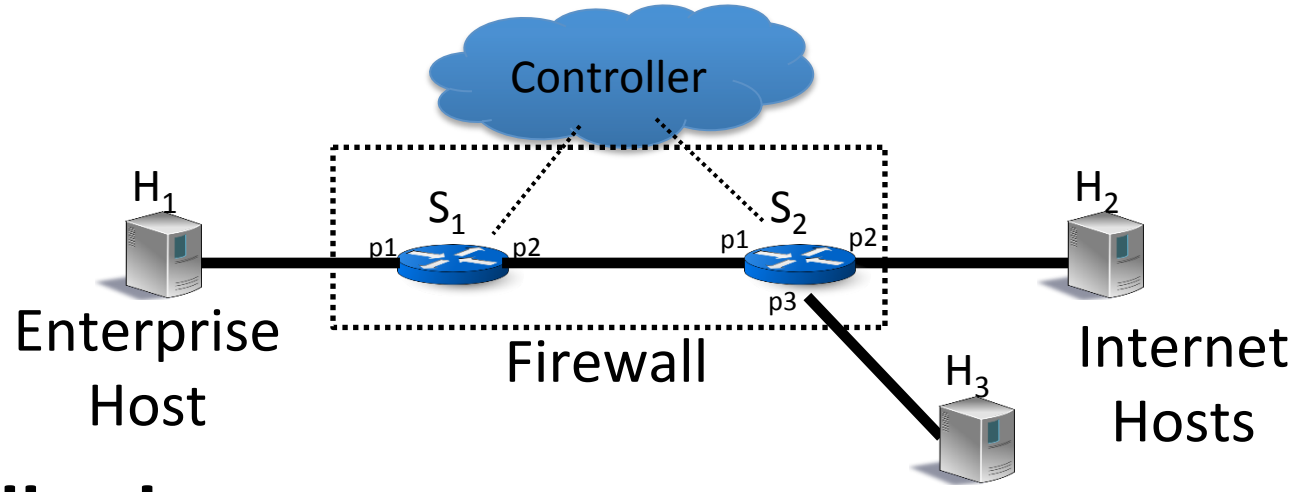
Full formal verification of Controller using model checking.

Extend model checking based approaches with abstractions to handle an unbounded number packets.

Overview

- Existing approaches and problem statement
- **Abstraction on Stateful firewall**
- Experimental case studies
  - Stateful firewall
  - Learning switch
- Conclusions

Stateful Firewall



Firewall rules:

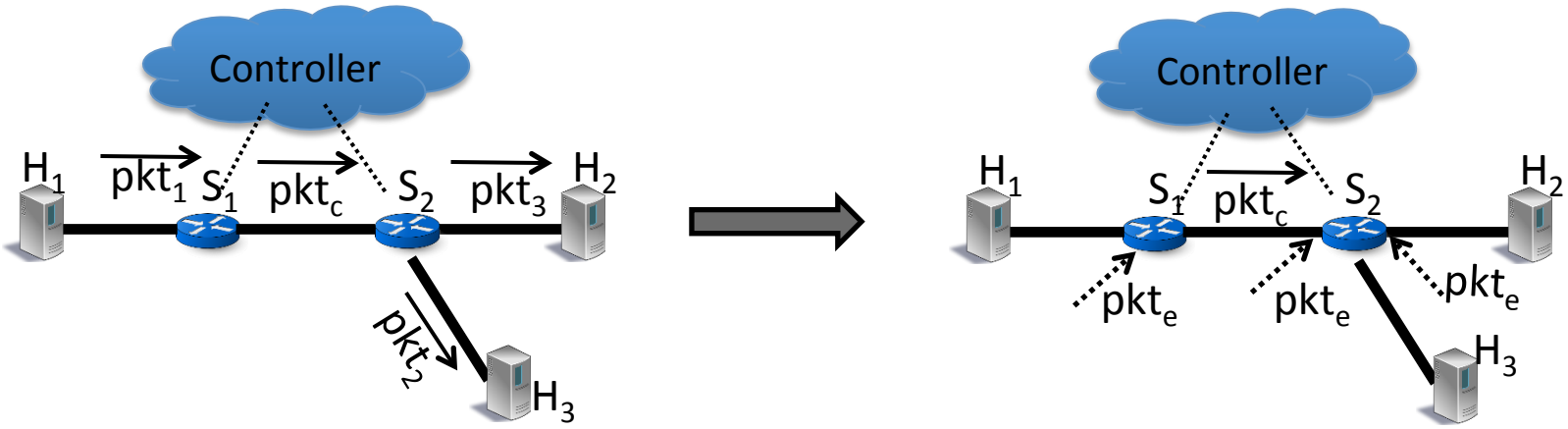
1) H_1 can contact H_2 or H_3.
2) H_2/H_3 can contact H_1, only if H_1 has already contacted them.
3) If H_2/H_3 initiates contact first, it must be blocked.

Property: If H_2 never contacts H_1 first, it does not get blocked.

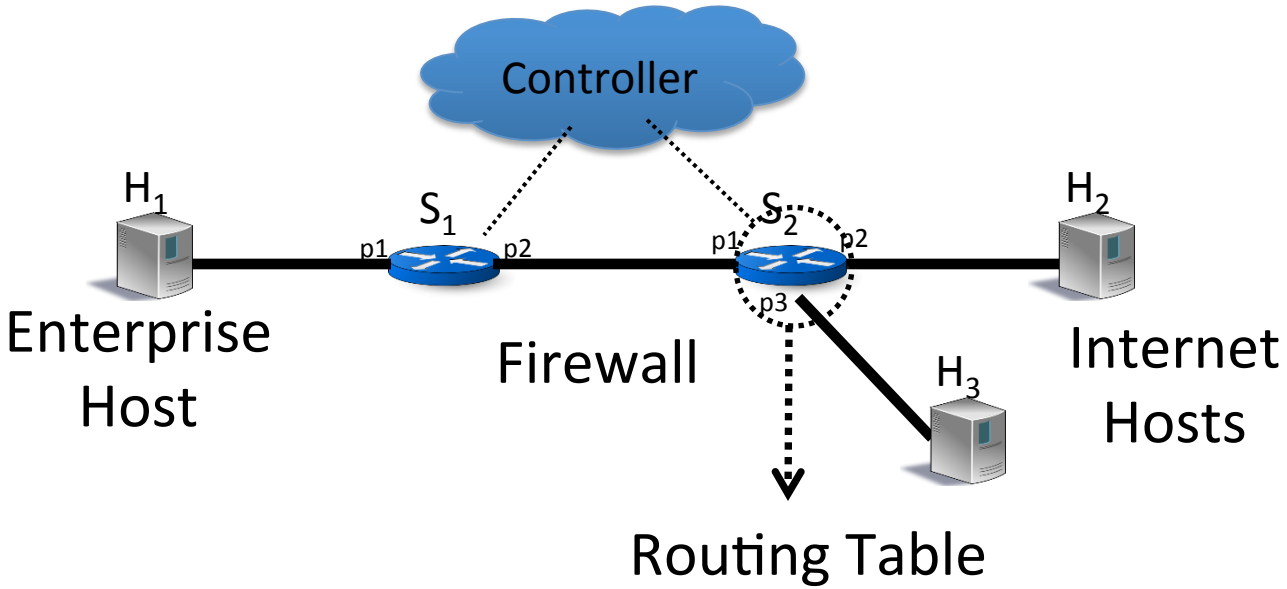
Abstraction for Unbounded Packets: Data State Abstraction

Key insight: properties of interest are per-packet properties.

- For example a packet from one host cannot reach another.



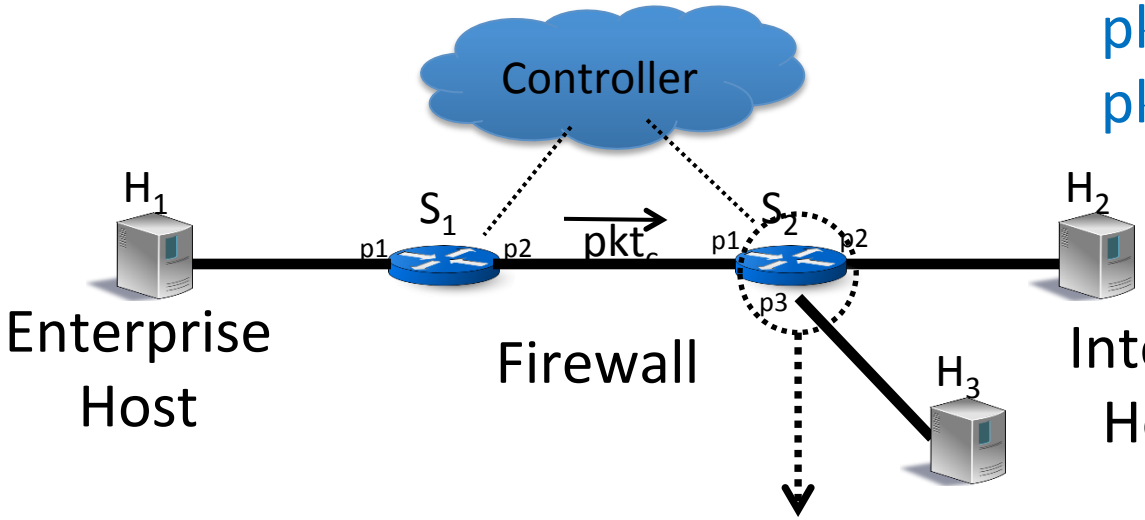
Abstraction for Large Switch State: Network State Abstraction



output port(pkt) =

- p_1: pkt.dst = H_1
- p_2: pkt. dst = H_2
- p_3: pkt. dst = H_3

Abstraction for Reducing Switch State: Leveraging Data State Abstraction



Abstracted Routing Table

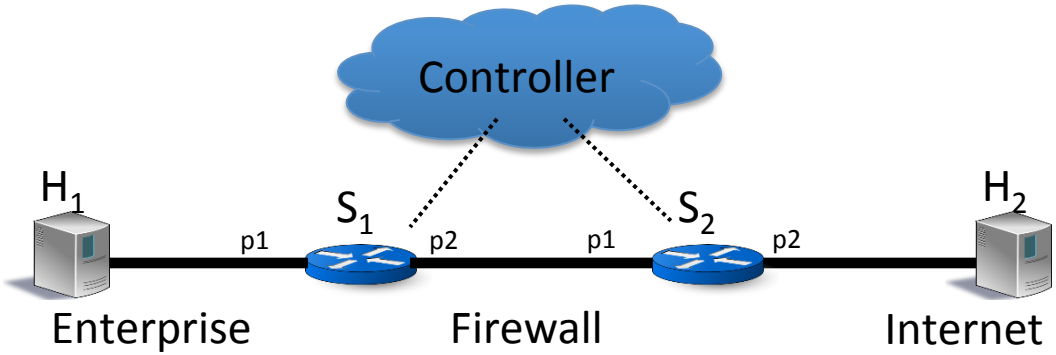
output port(pkt) =

- p_1: pkt.dst = H_1
- p_2: pkt. dst = H_2
- non-det: pkt. dst != {H_1 or H_2}

Overview

- Existing approaches and problem statement
- Abstraction on Stateful firewall
- **Experimental case studies**
  - Stateful firewall
  - Learning switch
- Conclusions

Stateful Firewall

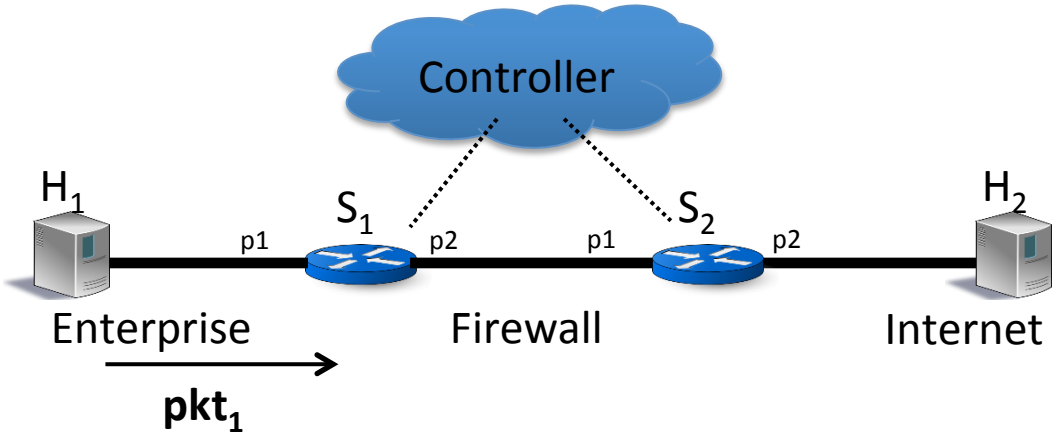


Verified a Murphi model of the firewall with a single host H_2.

- Found a bug: H_2 replies to H_1 but still gets blocked!

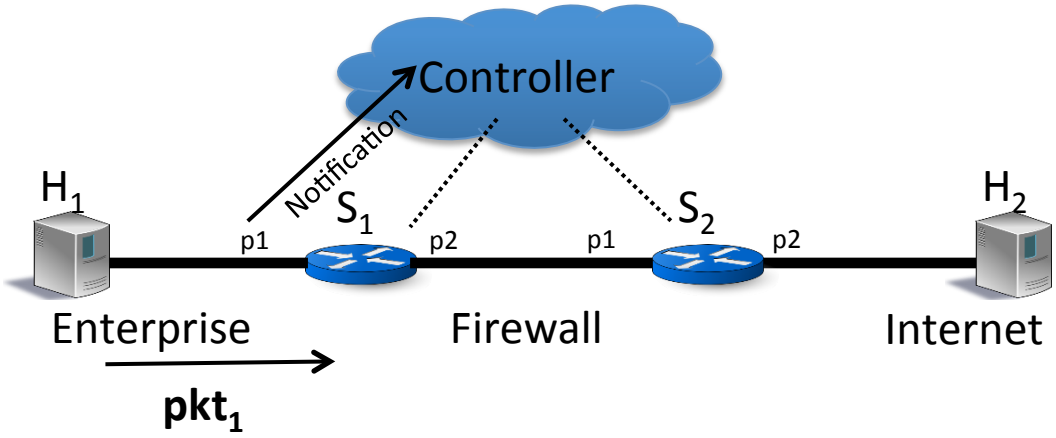
Experiments were done on a 2.40 GHz Intel Core 2 Quad processor, 3.74 GB RAM.

Stateful Firewall: Race Condition



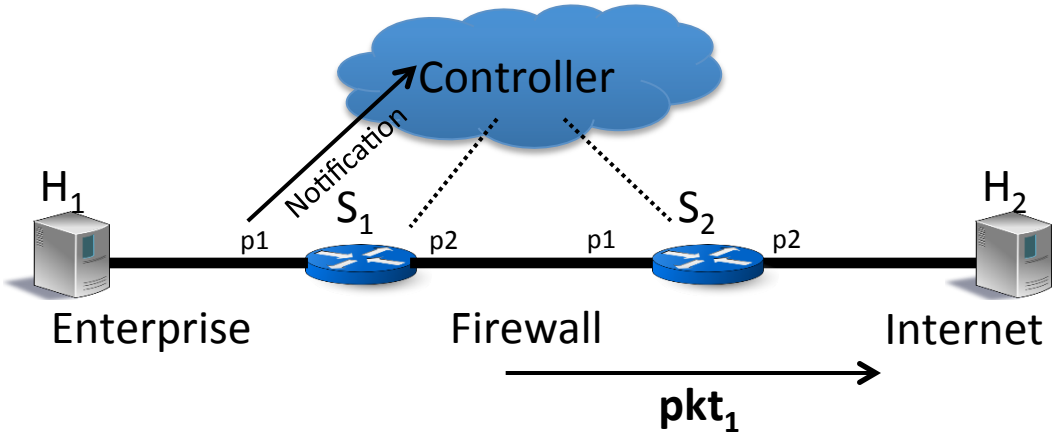
H_1 sends a packet pkt_1 to H_2

Stateful Firewall: Race Condition



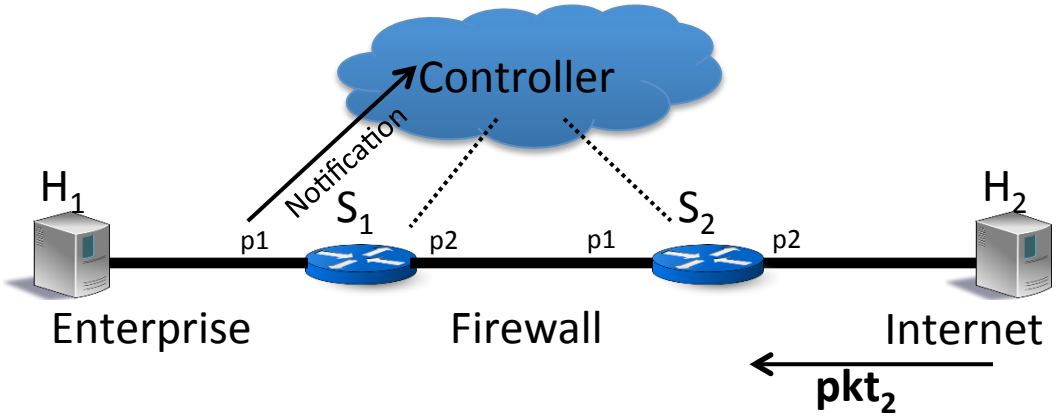
Switch S_1 notifies the controller.

Stateful Firewall: Race Condition



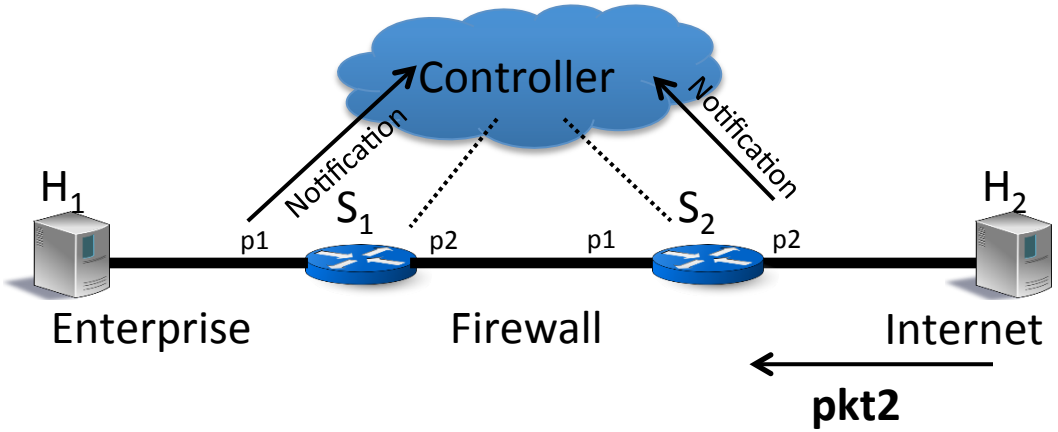
Packet is also forwarded by S_1, to S_2 which sends it to H_2.

Stateful Firewall: Race Condition



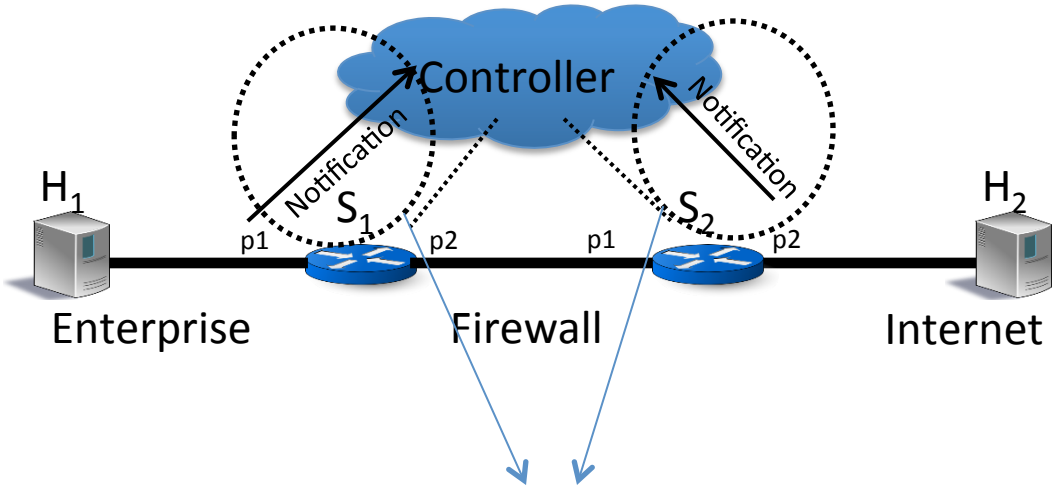
Host H_2 replies with packet pkt_2.

Stateful Firewall: Race Condition



Switch S_2 notifies Controller about pkt_2.

Stateful Firewall: Race Condition

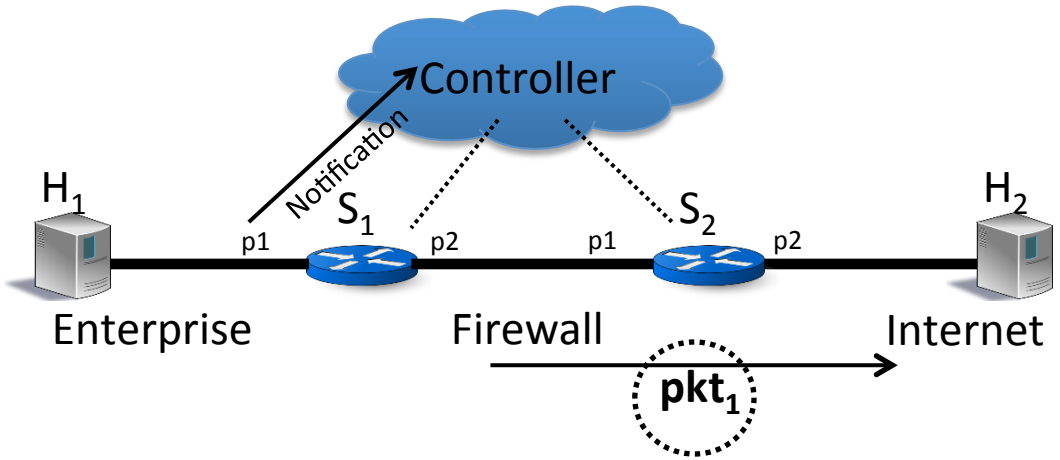


If notification of S_1 reaches after S_2, Controller thinks that H_2 contacted first and so is an attacker!

H_2 gets erroneously blocked!

Bug detected in 0.13 sec with 482 states

Stateful Firewall: Bug Fix

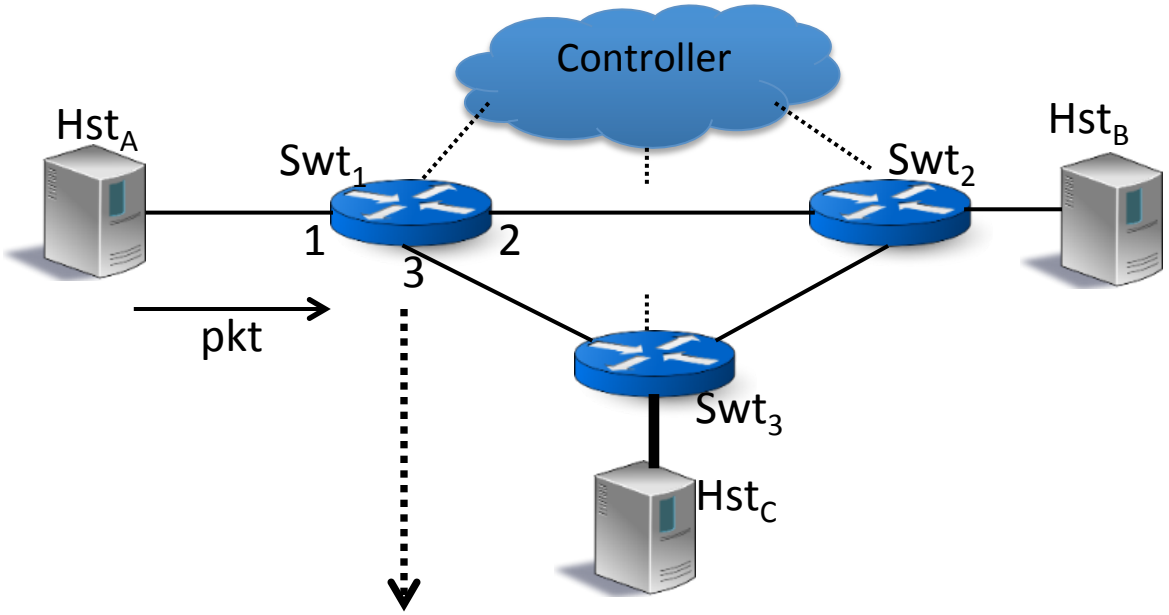


S_1 waits for Controller to acknowledge notification before forwarding packet pkt_1 to H_2.

- Proved correctness for an unbounded number of packets in this case.

Correctness proof for the bug free case with unbounded number of packets in 0.19 sec with 613 states

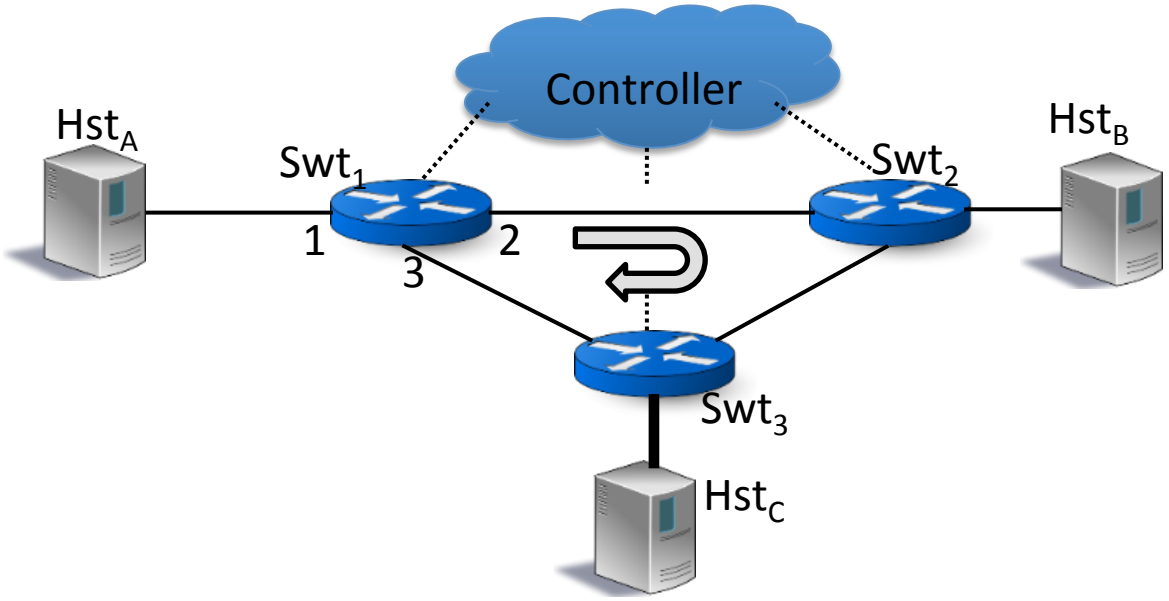
Learning Switch



When a packet arrives at a switch at an input port:

- Switch learns its source host is connected to that port.
- Uses this information to route future packets efficiently.

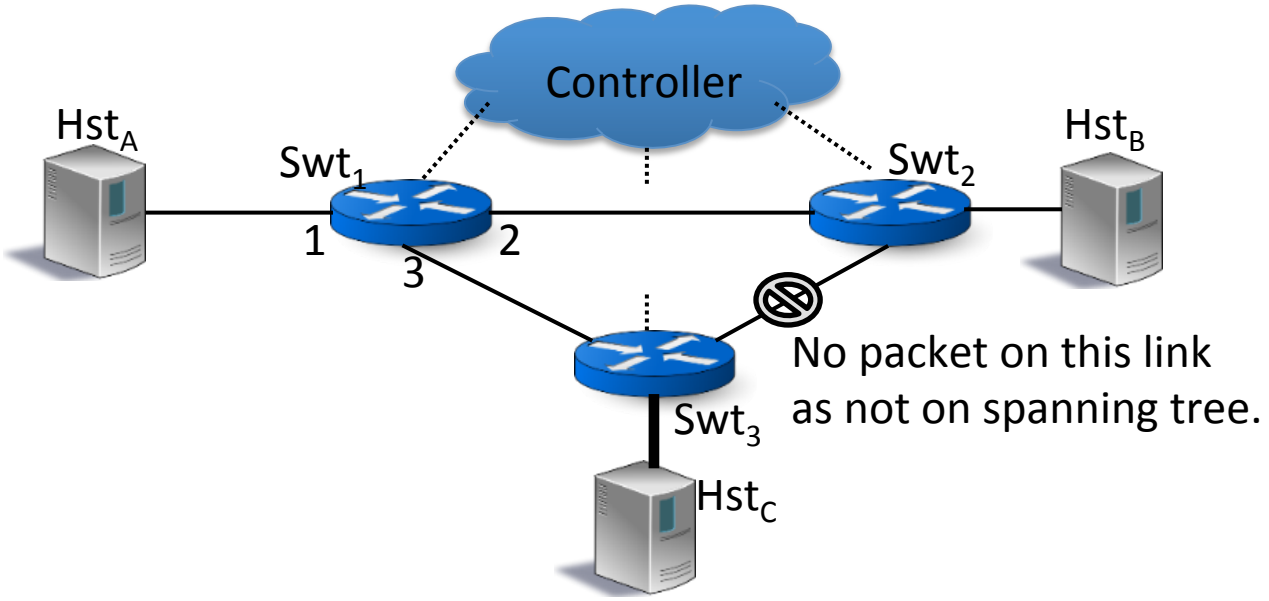
Learning Switch: Bug



Switches may learn routing information such that packets get stuck in a loop!

Loop was found in 0.1 sec with 159 states explored.

Learning Switch: Bug Fix



Only route on a spanning tree

Verified for an arbitrary number of packets exchanged between Hst$_A$ and Hst$_B$ in 600s with 1.45M.

Overview

- Existing approaches and problem statement
- Abstraction on Stateful firewall
- Experimental case studies
  - Stateful firewall
  - Learning switch
- **Conclusions**

Conclusions

- We presented abstractions for:
  - Verifying properties for an arbitrary number of packets.
  - Reducing network state.

- Verified a stateful firewall and a learning switch using these abstractions.

Thank You!

Stress test

- Stress test: Larger fat tree topology with 20 switches, 16 hosts and 48 links.
  - Model checking did not finish for an arbitrarily large number of packets.
  - It finished in 68352s for the single packet case with network state abstraction

Questions

- Lines of code?
- NAT ~1000
- Pyswitch ~1000
- Bug handled by acknowledgement carrying host info?