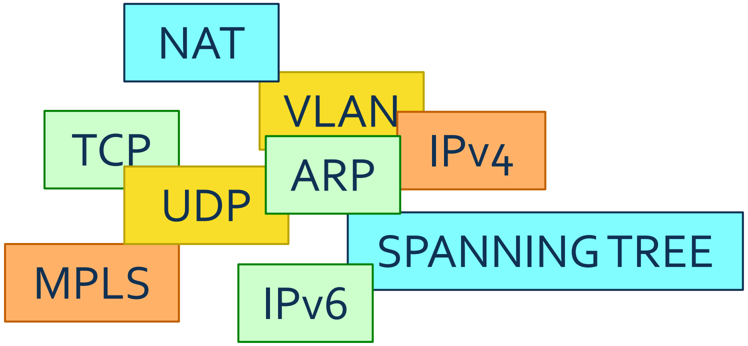
Header Space Analysis: Static Checking For Networks

Peyman Kazemian, Nick McKeown (Stanford University) and George Varghese (UCSD and Yahoo Labs).

Presented by Eviatar Khen (Software Defined Networks Seminar)

Today

- A typical network is a complex mix of protocols



- Interact in complex way
- Hard to understand, manage and predict the behavior

Today

- Even simple question are hard to answer:
  - “Can host A talk to host B?”
  - “What are all the packet headers from A that can reach B?”
  - “Can packets loop in my network?”
  - “Is Slice X isolated totally from Slice Y”?

Earlier Work

- Dynamic analysis: Veriflow (not mentioned in the paper)
- Static analysis: existing tools are protocol dependent, tailored to IP networks

Paper's Contribution

- Header Space Analysis – A general foundation that gives us:
  - A unified view of almost all types of boxes
  - An interface for answering different question about the network

Roadmap

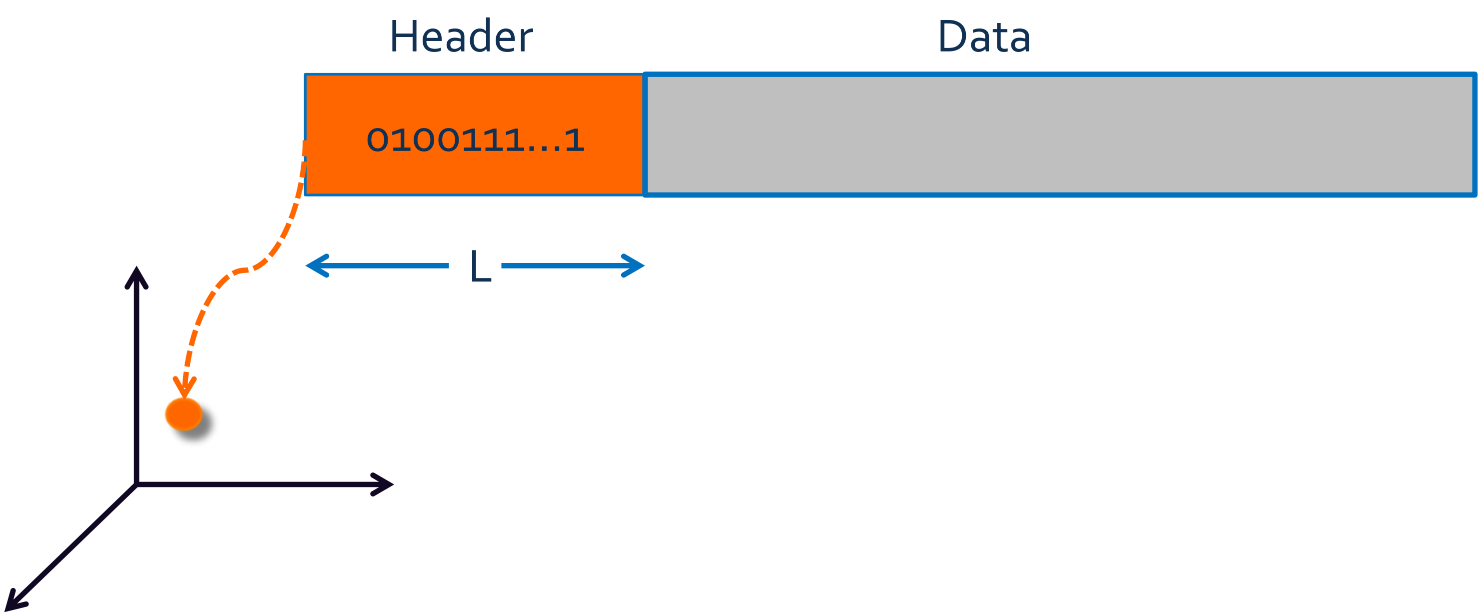
- The Header Space framework
- Use cases: how HSA could be used to detect network failures
- Experiments Results

Header Space Framework

Key observation: A packet is a point in a space of possible headers and a box is a transformer on that space

Header Space Framework

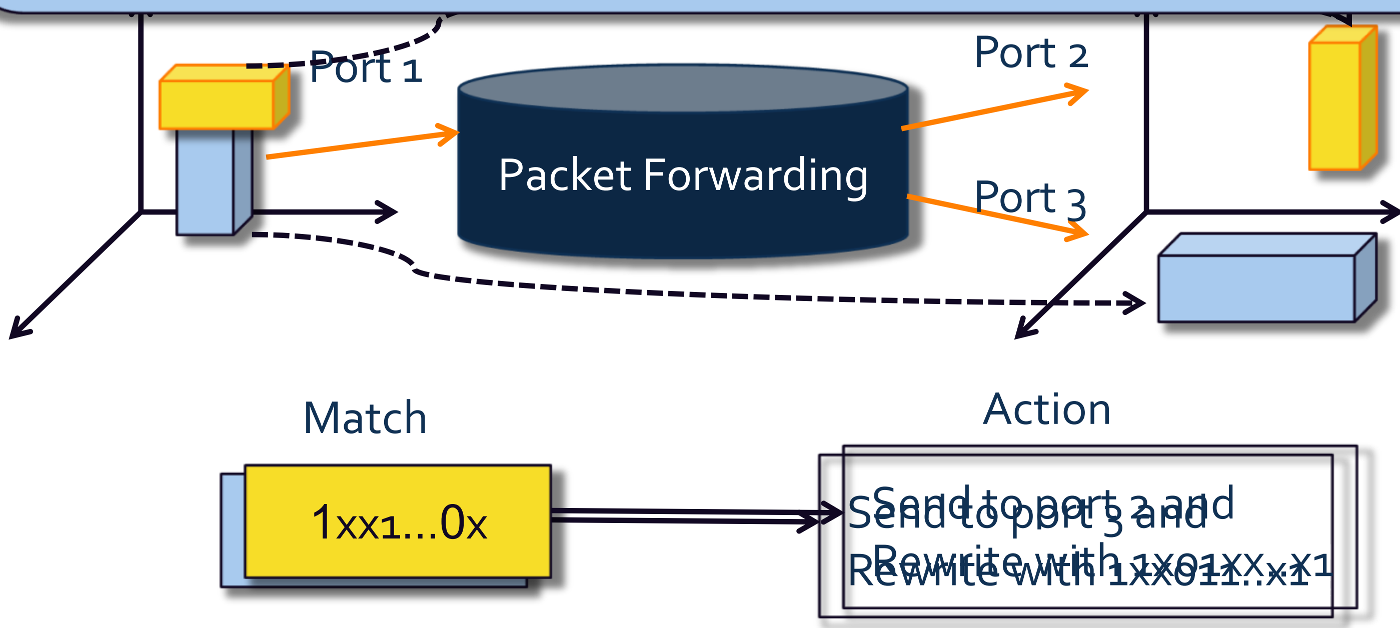
- Step 1: Model a Packet Header
- A Packet Header is a point in space $\{0,1\}^L$,called the <u>Header Space</u>



Header Space Framework

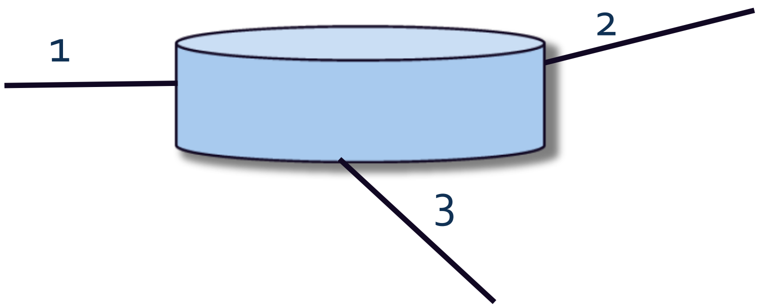
Transfer Function:

$$T(h,p):(h,p)\rightarrow\{(h_1,p_1),(h_2,p_2)\ldots\}$$



Header Space Framework

+ Example: Transfer Function of an IPv4 Router



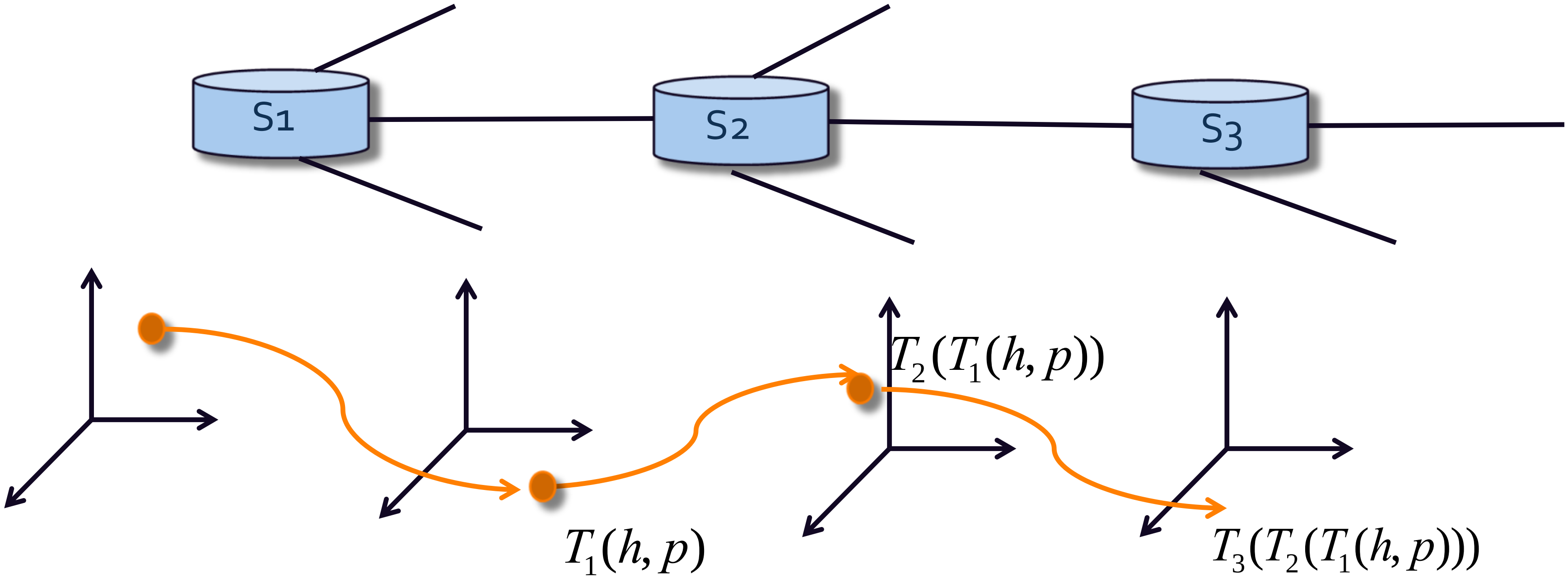
172.24.74.0, 255.255.255.0	Port 1
172.24.128.0, 255.255.255.0	Port 2
171.67.0.0, 255.255.0.0	Port 3

$$T(h,p) = \begin{cases} (h,1) & \text{if } dest\_ip(h) = 172.24.74.X \\ (h,2) & \text{if } dest\_ip(h) = 172.24.128.X \\ (h,3) & \text{if } dest\_ip(h) = 172.67.X.X \end{cases}$$

$$T(h,p) = \begin{cases} (dec\_ttl(h),1) & \text{if } dest\_ip(h) = 172.24.74.X \\ (dec\_ttl(h),2) & \text{if } dest\_ip(h) = 172.24.128.X \\ (dec\_ttl(h),3) & \text{if } dest\_ip(h) = 172.67.X.X \end{cases}$$

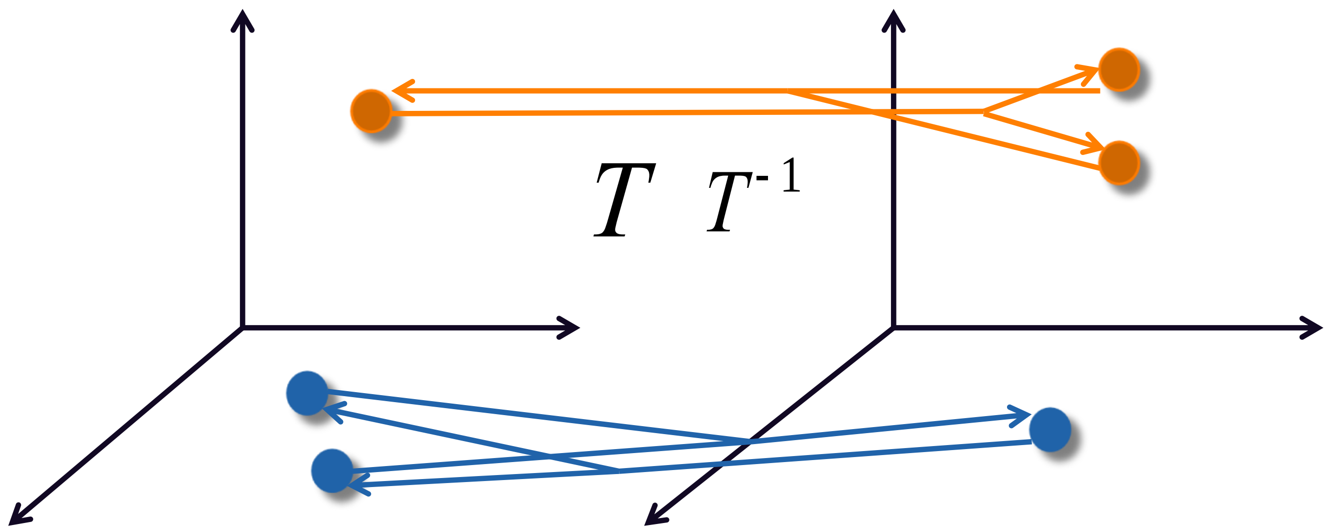
Header Space Framework

- Transfer Function Properties:
  - Composable: $T_3(T_2(T_1(h,p)))$



Header Space Framework

- Transfer Function Properties:
  - Invertible:



Doman (input) Range (output)

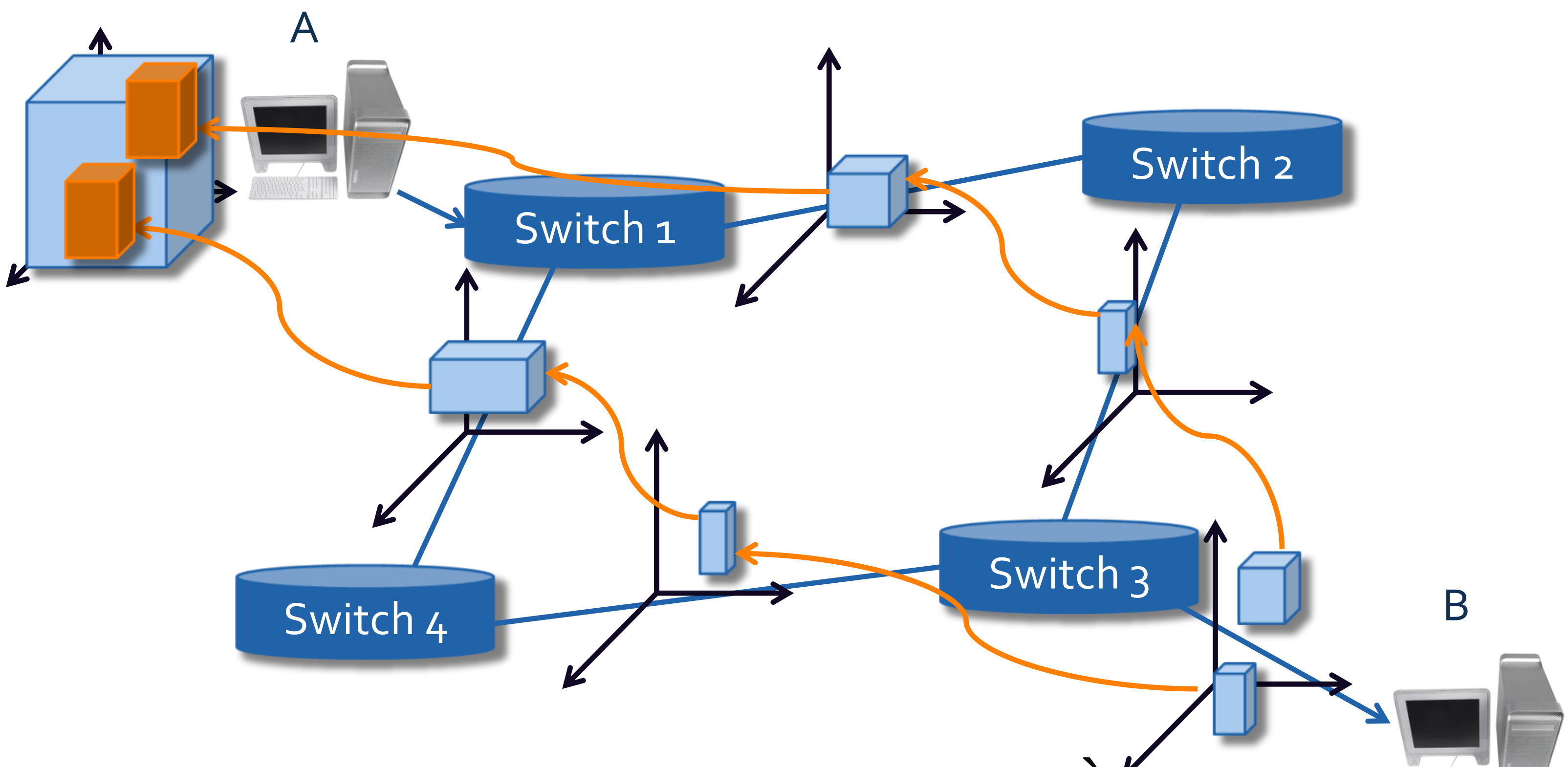
Header Space Framework

- Step 3: Develop an Algebra to work on these spaces
- A subspace correspond to a Wildcard
- We use this to define set operations on Wildcards:
  - Intersection
  - Complementation
  - Difference

Use Cases of Header Space Framework

Use Cases

- *"Can host A talk to host B?"*



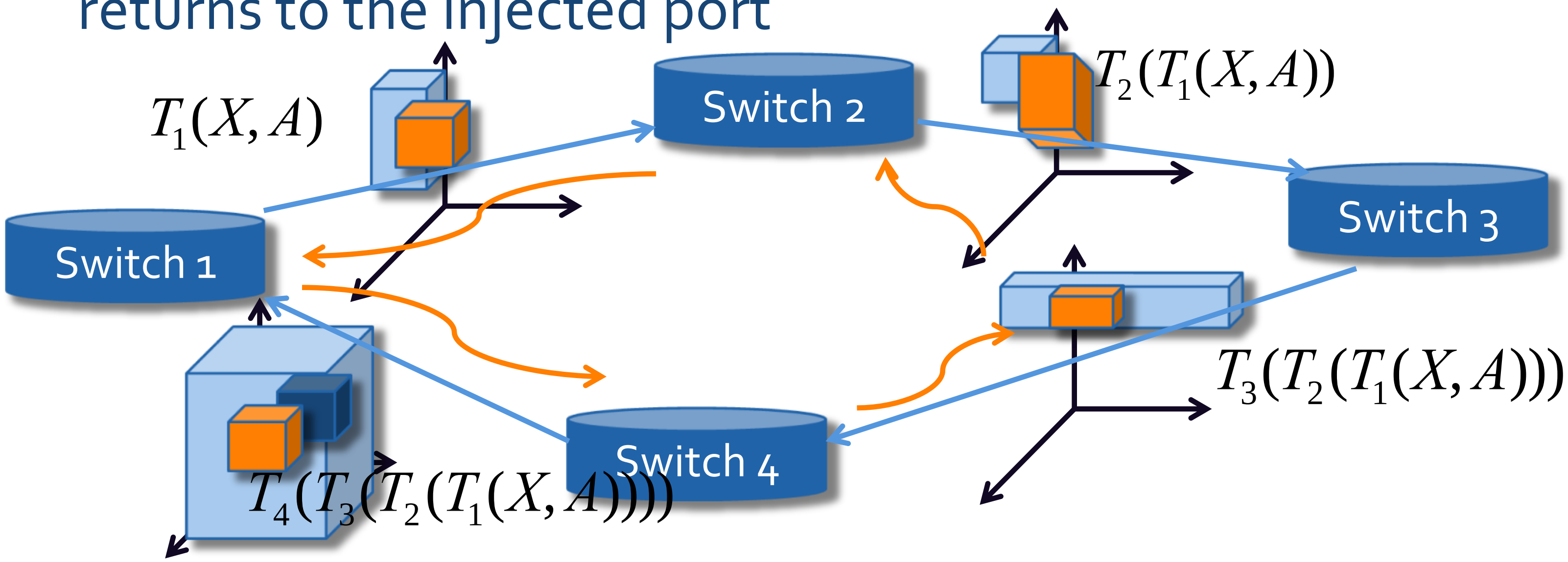
$$T_3(T_2(T_1(X,A))) \cup T_3(T_4(T_1(X,A)))$$

Complexity

- Each input wildcard is matched to each rule at the switch and creates an output wildcard
- So for R_1 inputs and R_2 rules the number of output wildcards can be $O(R_1 R_2)$
- In reality: Linear Fragmentation
- Overall: $O(dR^2)$ where d is the max diameter and R is the maximum number of rules

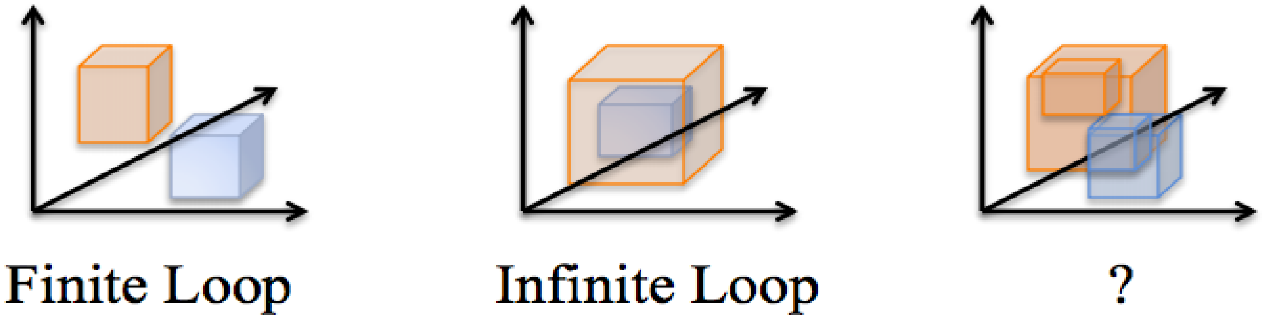
Use Cases

- "Is there a Loop in the Network?"
- Inject the whole header space from EACH port
- Follow the packet until it either leaves the network or returns to the injected port



Use Cases

- "Is the loop infinite?"

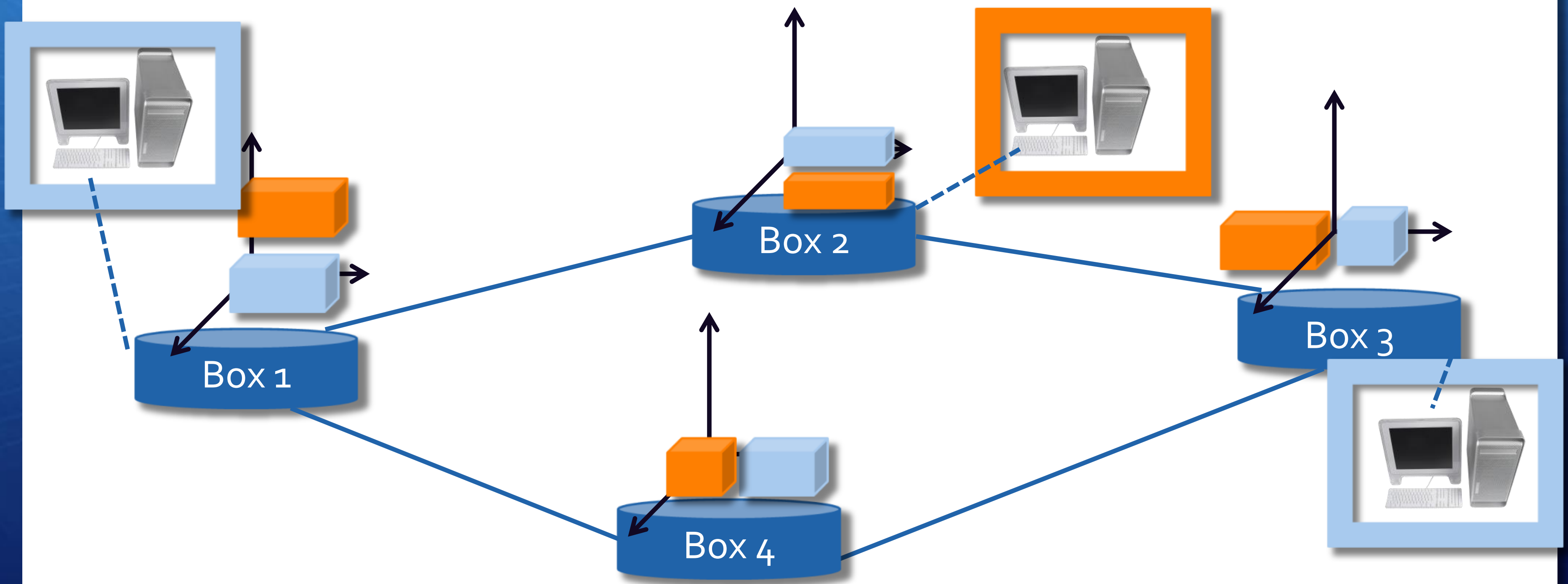


- Complexity O(dPR^2)

Use Cases

- Slicing a network is a way to share network resources among multiple entities
- Could be created using VLAN or FlowVisor
- Definition:
  - A topology consisting of switches, ports and links
  - a collection of predicates on packets belonging to the slice, one on each ingress port in the slice topology.
- "Are two Slices isolated?"

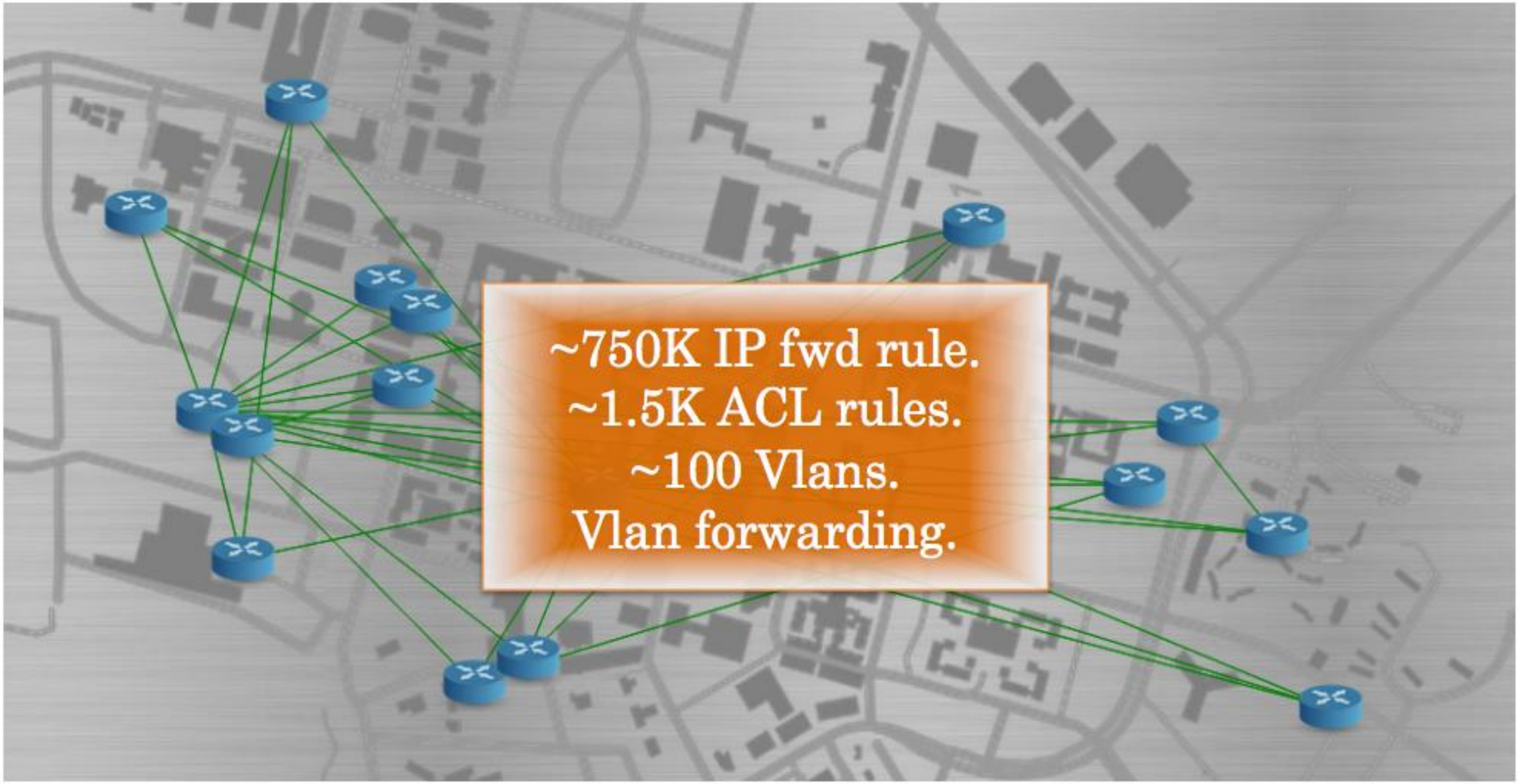
Use Cases



Implementation

- Header space Library (Hassel)
- Written in Python
- Header Space class
  - Encapsulates a union of wildcard expressions
  - Implements Set operations
- Transfer Function class
  - Implements T and T-1
- The Application allows: Reachability, Loop Detection and Slice Isolation checks.

Stanford backbone network



Performance

- Performance result for Stanford Backbone Network on a single machine: 2.66Ghz quad core, 4GB RAM

Generating TF Rules	~150 sec
Loop Detection Test (30 ports)	~560 sec
Average Per Port	~18 sec
Min Per Port	~ 8 sec
Max Per Port	~ 135 sec
Reachability Test (Avg)	~13 sec

Summary

- A General Foundation that gives us:
  - A unified view of almost all type of boxes:

  Transfer Function

  - An interface for answering different questions about the network

  T(h,p) and T-1(h,p)

  Set operations on Header Space

- The Python-based implementation can scale to enterprise-size networks on a single laptop