HEADER SPACE ANALYSIS

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

July 17th, 2012
Joint Techs 2012
Today...

- A typical network is a complex mix of protocols:
  - Interact in complex ways.
  - Cause unforeseen behavior.
  - Hard to manage, understand and predict the behavior of networks.
TODAY...

Even simple questions are hard to answer...

- Can host A talk to host B?
- What are all the packet headers from A that can reach B?
- Are there any loops in the network?
- Is Slice X isolated totally from Slice Y?
- What will happen if I remove an entry from a router?
HEADER SPACE ANALYSIS

- A Powerful **General Foundation** that gives us
  - A unified view of almost all type of boxes.
  - A powerful interface for answering different questions about the network.
HEADER SPACE FRAMEWORK

SIMPLE OBSERVATION: A PACKET IS A POINT IN THE SPACE OF POSSIBLE HEADERS AND A BOX IS A TRANSFORMER ON THAT SPACE.
Step 1 - Model packet header as a point in $\{0, 1\}^L$ space – The Header Space
**Header Space Framework**

- Step 2 – Model all networking boxes as transformer of header space

Transfer Function:

\[ T : (h_{in}, p_{in}) \rightarrow \{(h_1, p_1), (h_2, p_2), \ldots, (h_n, p_n)\} \]
**HEADER SPACE FRAMEWORK**

- **Example: Transfer Function of an IPv4 Router**

  - 172.24.74.0    255.255.255.0    Port1
  - 172.24.128.0   255.255.255.0    Port2
  - 171.67.0.0     255.255.0.0      Port3

\[
T(h, p) = \begin{cases} 
(h,1) & \text{if } \text{dst}_\text{ip}(h) = 172.24.74.x \\
(h,2) & \text{if } \text{dst}_\text{ip}(h) = 172.24.128.x \\
(h,3) & \text{if } \text{dst}_\text{ip}(h) = 171.67.x.x
\end{cases}
\]
HEADER SPACE FRAMEWORK

Example: Transfer Function of an IPv4 Router

- 172.24.74.0  255.255.255.0  Port1
- 172.24.128.0  255.255.255.0  Port2
- 171.67.0.0  255.255.0.0  Port3

\[
T(h, p) = \begin{cases} 
\text{(dec\_ttl}(h),1) & \text{if } \text{dst\_ip}(h) = 172.24.74.x \\
\text{(dec\_ttl}(h),2) & \text{if } \text{dst\_ip}(h) = 172.24.128.x \\
\text{(dec\_ttl}(h),3) & \text{if } \text{dst\_ip}(h) = 171.67.x.x
\end{cases}
\]
**Header Space Framework**

- **Example: Transfer Function of an IPv4 Router**

- 172.24.74.0 255.255.255.0 Port1
- 172.24.128.0 255.255.255.0 Port2
- 171.67.0.0 255.255.0.0 Port3

\[
T(h, p) = \begin{cases} 
(rw\_mac(\text{dec\_ttl}(h),\text{next\_mac}), 1) & \text{if } \text{dst\_ip}(h) = 172.24.74.x \\
(rw\_mac(\text{dec\_ttl}(h),\text{next\_mac}), 2) & \text{if } \text{dst\_ip}(h) = 172.24.128.x \\
(rw\_mac(\text{dec\_ttl}(h),\text{next\_mac}), 3) & \text{if } \text{dst\_ip}(h) = 171.67.x.x 
\end{cases}
\]
Properties of transfer functions

- Composable: \( T_3(T_2(T_1(h, p))) \)

- Invertible: \( T^{-1}T \)

Domain (input) \hspace{2cm} Range (output)
Step 3 - Develop an algebra to work on these spaces.

Every object in Header Space, can be described by union of Wildcard Expressions.

We want to perform the set operations on these wildcard expressions:

- Intersection
- Complementation
- Difference

Refer to “header space analysis” paper for details.
USE CASES OF HEADER SPACE FRAMEWORK

These are only some example use cases that we developed so far…
USE CASES

Can host A talk to B?

All Packets that A can use to communicate with B

\[ T^{-1}_1 \]

\[ T_1(X,A) \]

\[ T^{-1}_2 \]

\[ T_2(T_1(X,A)) \]

\[ T^{-1}_3 \]

\[ T_3(T_2(T_1(X,A))) \]

\[ T^{-1}_4 \]

\[ T_4(T_1(X,A)) \]

\[ T_3(T_4(T_1(X,A))) \]

\[ T_3(T_2(T_1(X,A))) \cup T_3(T_4(T_1(X,A))) \]
USE CASES

Is there a loop in the network?
- Inject an all-x text packet from every switch-port
- Follow the packet until it comes back to injection port
USE CASES

- Is the loop infinite?

Finite Loop

Infinite Loop

?
IMPLEMENTATION

- Header Space Library (Hassel)
  - Written in Python
  - Implements Header Space Class
    - Set operations
  - Implements Transfer Function Class
    - \( T \) and \( T^{-1} \)
  - Implements Reachability, Loop Detection and Slice Isolation checks.
    - < 50 lines of code
  - Includes a Cisco IOS and Juniper Junos parser
    - Generates transfer function from output of IOS/Junos commands and config file.
    - Keeps the mapping from Transfer function rule to line number in config file.
- Publicly available: git clone https://bitbucket.org/peymank/hassel-public.git
~750K IP fwd rule.
~1.5K ACL rules.
~100 Vlans.
Vlan forwarding.
Stanford Backbone Network

- Loop detection test – run time < 10 minutes on a single machine.

Vlan RED Spanning Tree

Vlan BLUE Spanning Tree
**Performance**

Performance result for Stanford Backbone Network on a single machine: 4 core, 4GB RAM.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Generating TF Rules</td>
<td>~150 sec</td>
</tr>
<tr>
<td>Loop Detection Test (30 ports)</td>
<td>~560 sec</td>
</tr>
<tr>
<td>Average Per Port</td>
<td>~18 sec</td>
</tr>
<tr>
<td>Min Per Port</td>
<td>~8 sec</td>
</tr>
<tr>
<td>Max Per Port</td>
<td>~135 sec</td>
</tr>
<tr>
<td>Reachability Test (Avg)</td>
<td>~13 sec</td>
</tr>
</tbody>
</table>
SUMMARY

- Header Space Analysis is a Powerful General Foundation that gives us
  - A unified view of almost all type of boxes.
  - A powerful interface for answering different questions about the network.
- Can be applied to both SDN and Non-SDN networks.
- Our initial Python-based implementation can scale to enterprise-size networks on a single machine.
Thank You!

Questions?
COMPLEXITY

Run time
- Reachability: $O(dR^2)$
- Loop Detection: $O(dPR^2)$
  - $R$: maximum number of rules per box.
  - $d$: diameter of network.
  - $P$: number of ports to be tested

See paper for more details.
**Complexity of Reachability and Loop Detection Tests**

- **Run time**
  - Reachability: $O(dR^2)$
  - Loop Detection: $O(dPR^2)$
  - $R$: maximum number of rules per box.
  - $d$: diameter of network.
  - $P$: number of ports to be tested

**Assumption**: Linear Fragmentation