HEADER SPACE ANALYSIS: STATIC CHECKING FOR NETWORKS

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

April 25th, 2012
NSDI 2012
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.
HEADER SPACE FRAMEWORK

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

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

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

  1. 172.24.74.0 255.255.255.0 Port1
  2. 172.24.128.0 255.255.255.0 Port2
  3. 171.67.0.0 255.255.0.0 Port3

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

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

- **Properties of transfer functions**
  - **Composable:** \( T_3(T_2(T_1(h, p))) \)
  - **Invertible:**

\[ T^{-1} \]

- **Domain (input)**
- **Range (output)**
**Header Space Framework**

- **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 following set operations on wildcard expressions:
  - Intersection
  - Complementation
  - Difference
Finding Intersection:

- Bit by bit intersect using intersection table:
  - Example: $10xx \cap 1xx0 = 10x0$
  - If result has any ‘$z$’, then intersection is empty:
  - Example: $10xx \cap 0xx1 = z0x1 = \phi$

See the paper for how to find complement and difference.
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

\[ T_1(X,A) \]

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

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

?
USE CASES

- Are two slices isolated?

- What do we mean by slice?
  - Fixed Slices: VLAN slices
  - Programmable Slices: slices created by FlowVisor

- Why do we care about isolation?
  - Banks: for added security.
  - Healthcare: to comply with HIPAA.
  - GENI: to isolate different experiments running on the same network.
**Use Cases**

- Are two slices isolated?
  - 1) slice definitions don’t intersect.
  - 2) packets do not leak.

Solution: Apply header space reservation of each slice to the slice’s transfer function and check intersection of the result with other slices’ reservations.
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

Slice Isolation Test: $O(NW^2)$
- $W$: number of wildcard expressions in definition of a slice.
- $N$: number of slices in the network.

See paper for more details.
**HEADER SPACE FRAMEWORK**

- A Powerful General Foundation that gives us
  - A unified view of almost all type of boxes.
    - Transfer Function.
  - A powerful interface for answering different questions about the network.
    - $T(h,p)$ and $T^{-1}(h,p)$
    - Set operations on Header Space
IMPLEMENTATION AND EVALUATION
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 parser
    - Generates transfer function from output of IOS 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
STANFORD BACKBONE NETWORK

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

<table>
<thead>
<tr>
<th>Test</th>
<th>Time</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>
We Introduced 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.

We showed that our initial Python-based implementation can scale to enterprise-size networks on a single laptop.
Thank You!

Questions?