Lecture 8 Network Verification

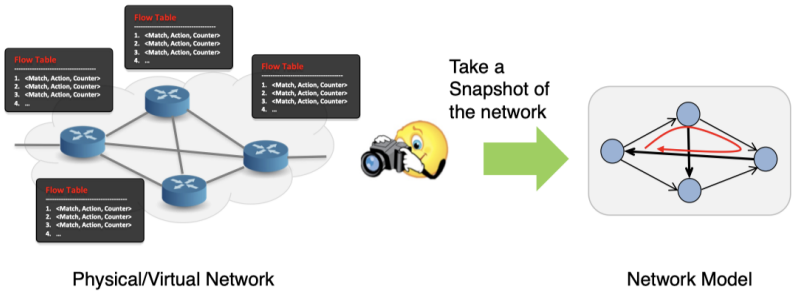
- Network outages (停电 / 宕机) are common
- Most network outages and performance issues result from misconfiguration
- Most are Human Factors

Software Defined Network (SDN)

1. Simplify the management and controlling of Networks
2. Bug: Controller / Applications
3. Some local problems may influence "total" Networks

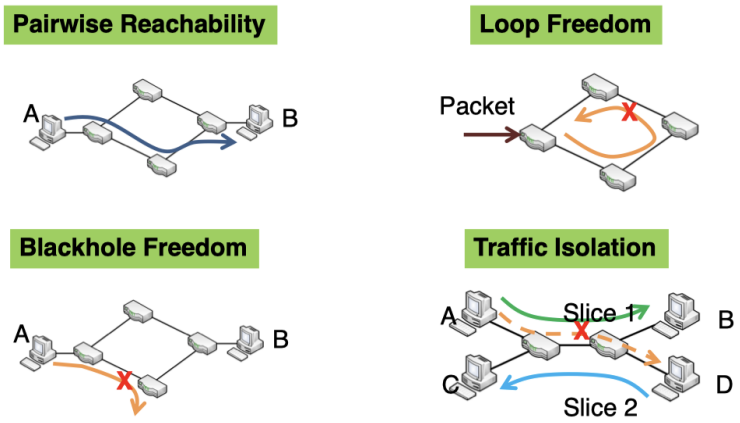
Data Plane Verification

1. Physical / Virtual Network (data / config table ……)
2. Take a snapshot of the network
3. Network Model



What to Verify

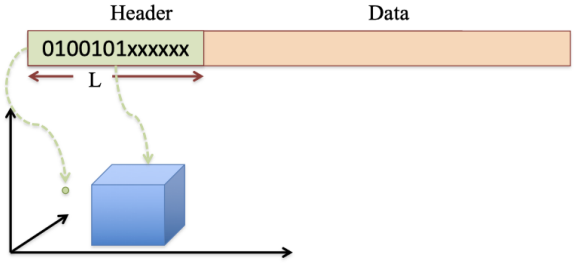
- Pairwise Reachability（可达性）
- Loop Freedom（环路检测）
- Blackhole Freedom（数据包陷进黑洞）
- Traffic Isolation（流量隔离性）



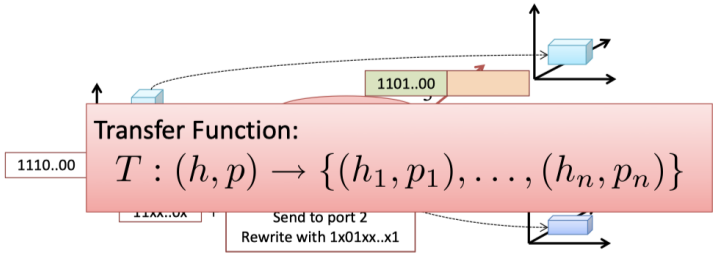
Header Space Analysis

1. Model a packet (header) as a point in $\{0,1\}^L$ space, i.e., the header space
    1. Flat space, Protocol oblivious
    2. *Wildcard Expression(通配符表达式)* (e.g., IP prefix) → hyper cubes
2. Model all networking boxes as transformers of header space
    1. Output = $f(Match, Action)$
    2. Transfer Function: $T : (h, p) => \{(h_1, p_1), \ldots, (h_n, p_n)\}$

- Step 1 – Model a packet (header) as a point in $\{0,1\}^L$ space, i.e., the header space
  - Flat space, protocol oblivious
  - Wildcard expression (e.g., IP prefix) -> hyper cubes

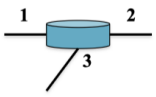


- Step 2 – Model all networking boxes as transformers of header space



- IPv4 Router – forwarding + TTL + MAC rewrite

  - 172.24.74.x Port1
  - 172.24.128.x Port2
  - 171.67.x.x Port3



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

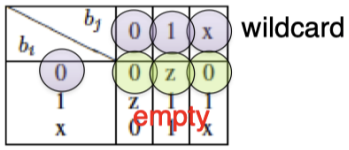
Theorems

- Composition Theorem
  - Network Behavior = Composition of transfer functions
  - e.g., $T_3(T_2(T_1(h,p)))$
- Inversion Theorem
  - given header h at destination p, we can invert to find (h',p'): headers sent at source s' to produce (h,p)

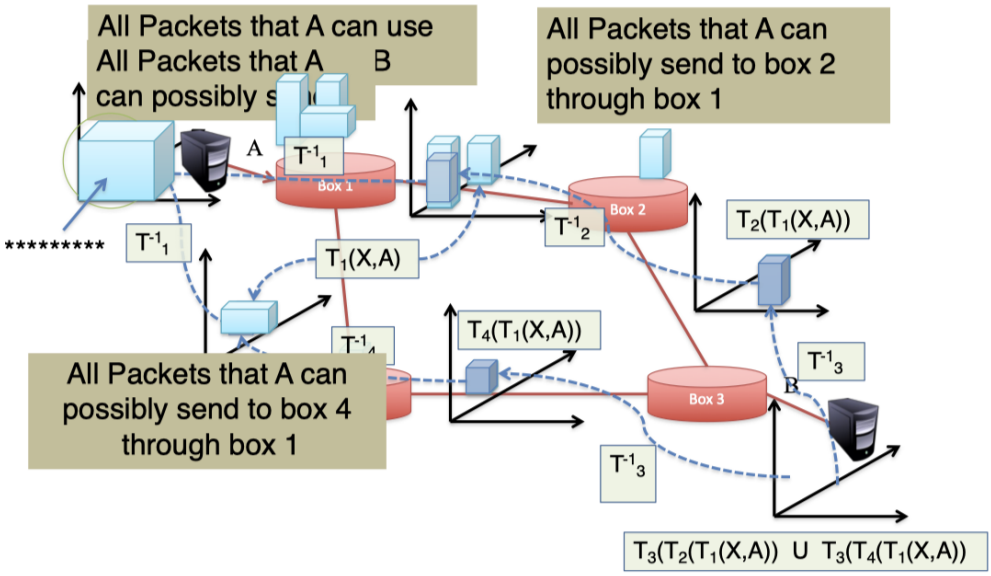
HSA Algebra

Bit by bit intersect using intersection (路口) table:

- Example: $10xx \cap 1xx0 = 10x0$
- If result has any 'z', then intersection is empty:
- Example: $10xx \cap 0xx0 = z0x0 = \phi$



Case 1: Computing Reachability

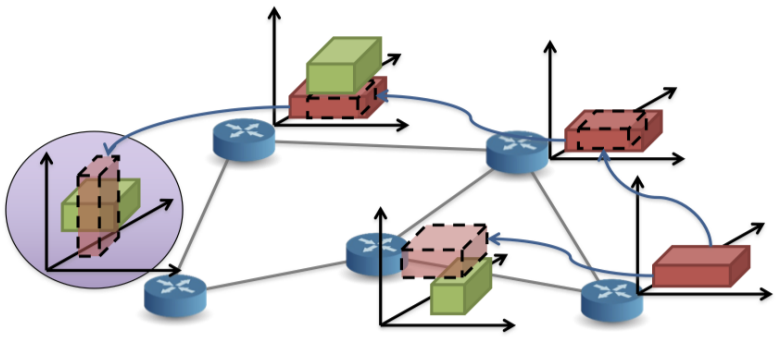


Case 2: Checking Isolation

切片

- Slice definitions don't intersect.
- Packets don't leak after forwarding.

泄漏



Problems with HSA

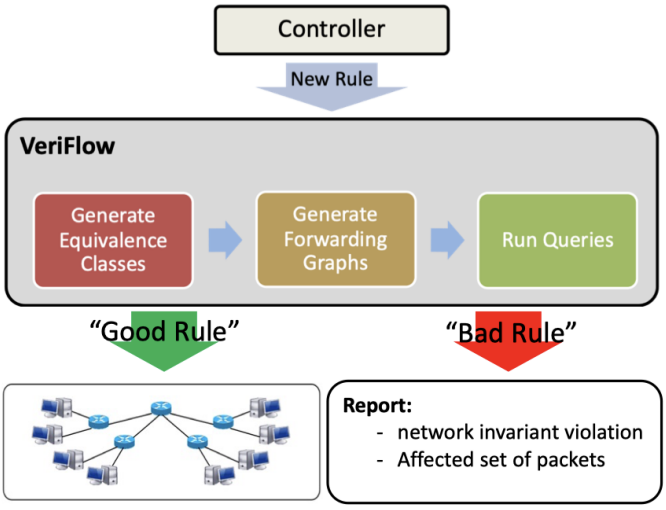
- Only check a snapchat of network configuration
- Networks are dynamically changing
- What if a new rule is inserted?
  - checking the entire network's state every time a new flow is wasteful and slow

What's VeriFlow?

Verifying Network Wide Invariants in **Real Time**

Controller set new rule:

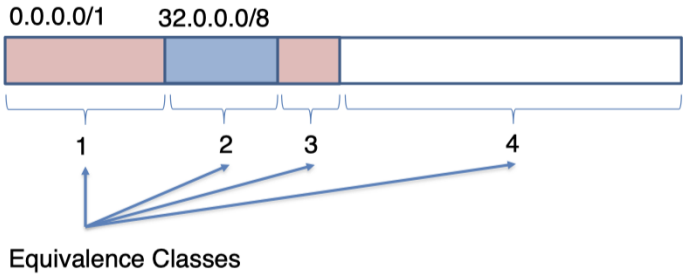
- Generate Equivalence Classes
- Generate Forwarding Graphs
- Run Queries

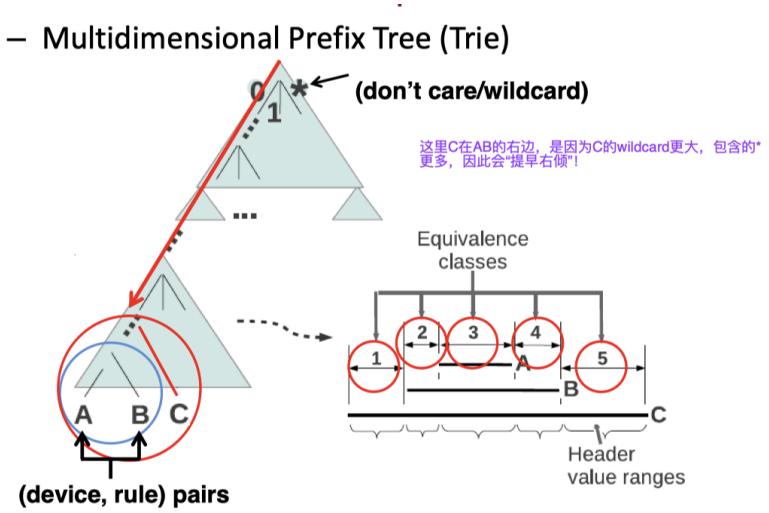


Challenging: Huge space of headers, impossible to enumerate

Step1: Generate Equivalence Classes

- Equivalence Class（等价类）
  - packets experiencing the *same forwarding actions* throughout the network
  - 一个等价类，不管大小多少，其内部的转发行为是一模一样的
- Data Structure for EC computation
  - multidimensional Prefix Tree (Trie-Tree)
  - each branch in a node is "0/1/*", we don't care "wildcard"
  - (device, rule) pairs
  - the tree represents the "total" Network

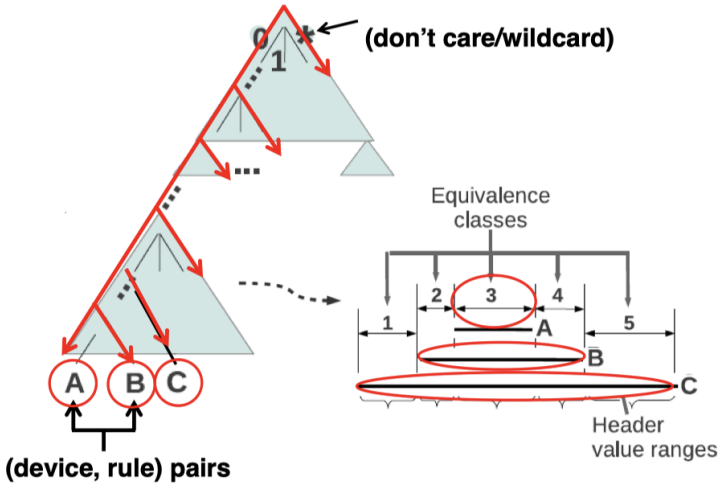




- C在AB的右侧，因为C的"匹配域"更长，说明"*"的含量更高，因此偏右!
- A / B / C 可能并不是一个交换机上的规则

Step2: Generate Forwarding Graphs

- Generate a forwarding graph for each EC
  - Define how packets within that EC will be forwarded through the network
  - Node: device, edge: forwarding rule
  - The graph will be "one-direction" between two nodes
  - Each node represents one EC
- Let the EC traverse the Trie for the second time



source code

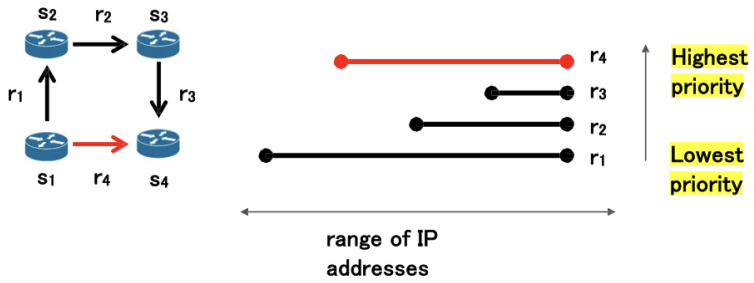
Step3: Run Queries (Verifying)

Check whether the forwarding graph of each EC satisfies

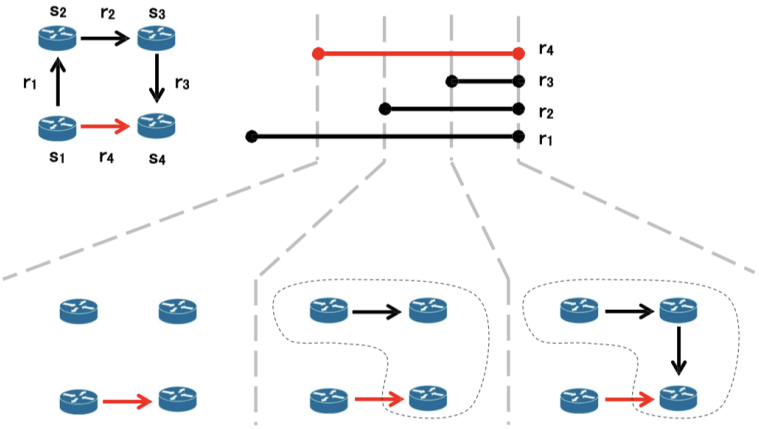
- Reachability
- Loop-freedom
- Blackhole-freedom

Analysis: Inefficiency of VeriFlow

1. High → Low Priority



2. Matching

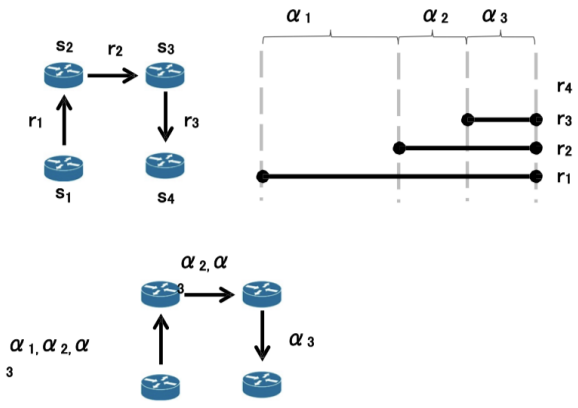


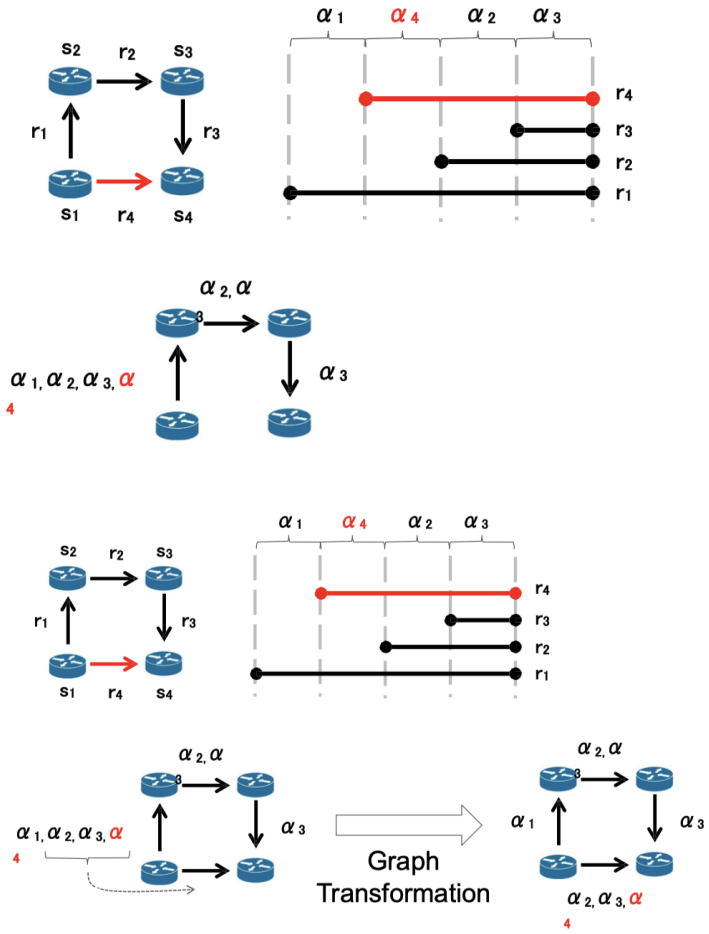
上面可以很清晰地看出一个问题：在不同的等价类中，路径有很多重叠部分

Delta-net

Rather than re-computing forwarding graphs, it **incrementally** maintains a single edge-labelled graph! represents all packet flows.

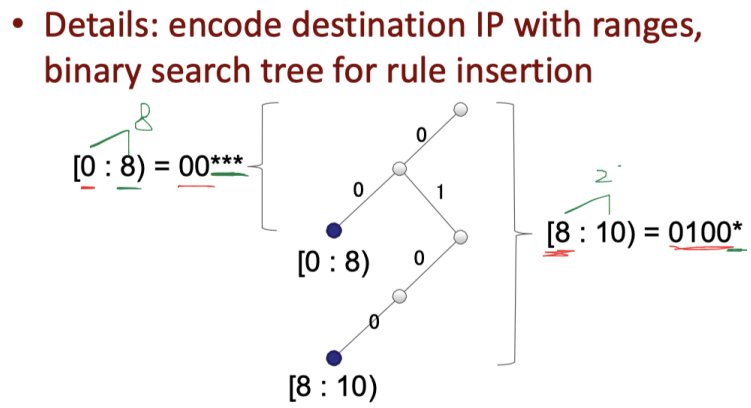
操作原理及其示范：





- 在上面的优先级更高
- alpha2,3,4 从 r1 边 移动到 r4 边
- 表示转发行为的变化（Graph Transformation）

Details: encode destination IP with ranges, binary search tree for rule insertion



Limitations

- Only considers a single dimention, i.e., destination IP
- Only works for IP ranges, e.g., IP prefixes

- The # of atoms os not minimum