Verificação formal em Redes Definidas por Software

Universidade Federal do Rio Grande do Sul

Autor: Levindo Gabriel Taschetto Neto (levindo.neto@inf.ufrgs.br)
Orientador: Alberto Egon Schaeffer-Filho (alberto@inf.ufrgs.br)

Introdução

Atualmente a necessidade por mais programabilidade e flexibilidade encoraja a utilização de abstrações de software para realizar atividades relacionadas à operação da rede, como por exemplo, algoritmos de roteamento e balanceadores de carga.

Concomitante, a centralização do plano de controle torna-se necessária para flexibilizar o gerenciamento da rede, principalmente no que diz respeito à lógica de encaminhamento de pacotes executada por *switches* e roteadores. Um conceito que visa centralizar o plano de controle da rede, e permitir sua programabilidade, é o de redes definidas por software (ou *Software Defined Networking (SDN)*). *SDN* é uma abordagem que propõe separar o plano de controle e os dispositivos de encaminhamento de dados.

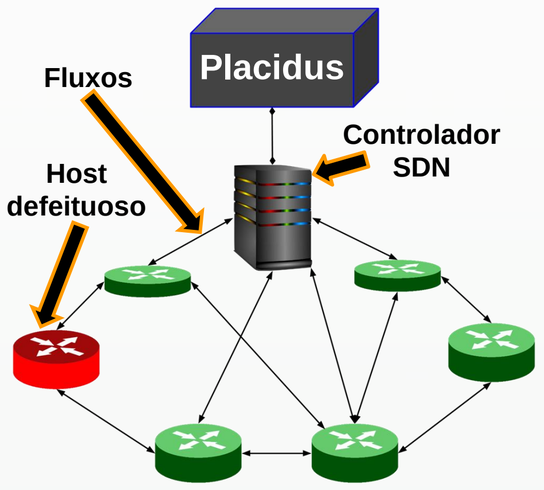
Foco do Trabalho

O foco do presente trabalho é investigar métodos e desenvolver aplicações para verificar formalmente, em tempo real, as seguintes propriedades em redes definidas por software:

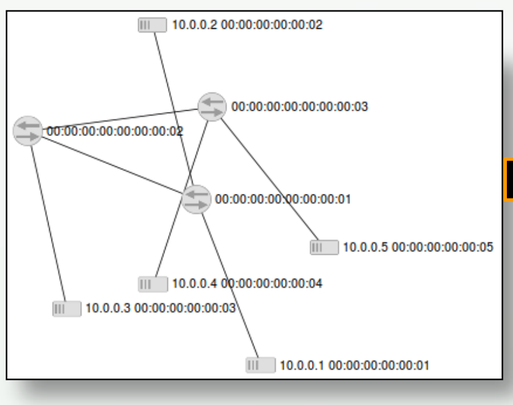
Conflitos:
Regras com **mesmo *match*** (dados de reconhecimento de pacote), mas **ações diferentes**.

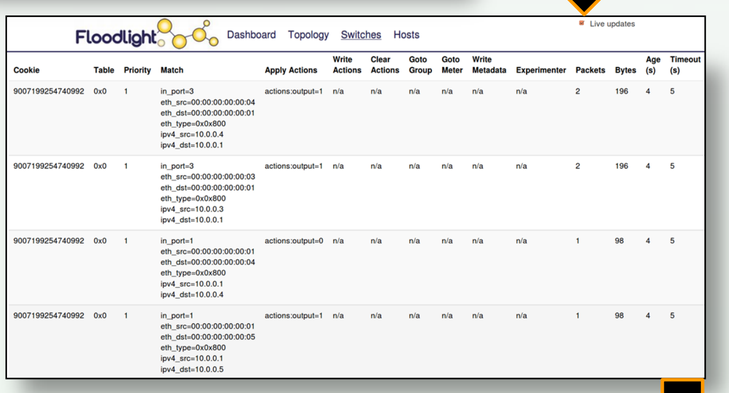
Redundâncias:
Regras com **mesmo *match*** (dados de reconhecimento de pacote) e **ações iguais**.

***Reachability*:**
Determinado **pacote** que sai de um *switch* A **chega** em um *switch* B de maneira certa.



Coleta de Dados




Floodlight Dashboard Topology Switches Hosts — Live updates

Cookie	Table	Priority	Match	Apply Actions	Write Actions	Clear Actions	Goto Group	Goto Meter	Write Metadata	Experimenter	Packets	Bytes	Age (s)	Timeout (s)
9007199254740992	0x0	1	in_port=3 eth_src=00:00:00:00:00:04 eth_dst=00:00:00:00:00:01 eth_type=0x0x800 ipv4_src=10.0.0.4 ipv4_dst=10.0.0.1	actions:output=1	n/a	n/a	n/a	n/a	n/a	n/a	2	196	4	5
9007199254740992	0x0	1	in_port=3 eth_src=00:00:00:00:00:03 eth_dst=00:00:00:00:00:01 eth_type=0x0x800 ipv4_src=10.0.0.3 ipv4_dst=10.0.0.1	actions:output=1	n/a	n/a	n/a	n/a	n/a	n/a	2	196	4	5
9007199254740992	0x0	1	in_port=1 eth_src=00:00:00:00:00:01 eth_dst=00:00:00:00:00:04 eth_type=0x0x800 ipv4_src=10.0.0.1 ipv4_dst=10.0.0.4	actions:output=0	n/a	n/a	n/a	n/a	n/a	n/a	1	98	4	5
9007199254740992	0x0	1	in_port=1 eth_src=00:00:00:00:00:01 eth_dst=00:00:00:00:00:05 eth_type=0x0x800 ipv4_src=10.0.0.1 ipv4_dst=10.0.0.5	actions:output=1	n/a	n/a	n/a	n/a	n/a	n/a	1	98	4	5

['1'	'2'	'42'	'1'	'9007199254740992'	'00:00:00:00:00:04'	'00:00:00:00:00:02'	'0x0x806'	'3'	'0'
['1'	'2'	'0'	'0'	'9007199254740992'	'00:00:00:00:00:02'	'00:00:00:00:00:01'	'0x0x806'	'2'	'1'
['1'	'2'	'42'	'1'	'9007199254740992'	'00:00:00:00:00:05'	'00:00:00:00:00:01'	'0x0x806'	'3'	'1'
['1'	'2'	'42'	'1'	'9007199254740992'	'00:00:00:00:00:05'	'00:00:00:00:00:02'	'0x0x806'	'3'	'1'
['1'	'2'	'0'	'0'	'9007199254740992'	'00:00:00:00:00:03'	'00:00:00:00:00:02'	'0x0x806'	'3'	'0'
['1'	'2'	'42'	'1'	'9007199254740992'	'00:00:00:00:00:04'	'00:00:00:00:00:01'	'0x0x806'	'3'	'1'
['1'	'2'	'0'	'0'	'9007199254740992'	'00:00:00:00:00:03'	'00:00:00:00:00:01'	'0x0x806'	'3'	'0'
['1'	'2'	'196'	'2'	'9007199254740992'	'00:00:00:00:00:04'	'00:00:00:00:00:01'	'0x0x800'	'3'	'0'
['1'	'2'	'98'	'1'	'9007199254740992'	'00:00:00:00:00:03'	'00:00:00:00:00:01'	'0x0x800'	'3'	'1'
['1'	'2'	'98'	'1'	'9007199254740992'	'00:00:00:00:00:01'	'00:00:00:00:00:04'	'0x0x800'	'1'	'1'
['1'	'2'	'98'	'1'	'9007199254740992'	'00:00:00:00:00:01'	'00:00:00:00:00:05'	'0x0x800'	'1'	'1'
['1'	'2'	'98'	'1'	'9007199254740992'	'00:00:00:00:00:02'	'00:00:00:00:00:04'	'0x0x800'	'2'	'0'
['1'	'2'	'98'	'1'	'9007199254740992'	'00:00:00:00:00:03'	'00:00:00:00:00:02'	'0x0x800'	'3'	'1'
['1'	'2'	'196'	'2'	'9007199254740992'	'00:00:00:00:00:04'	'00:00:00:00:00:02'	'0x0x800'	'3'	'1'
['1'	'2'	'98'	'1'	'9007199254740992'	'00:00:00:00:00:02'	'00:00:00:00:00:01'	'0x0x800'	'2'	'1'

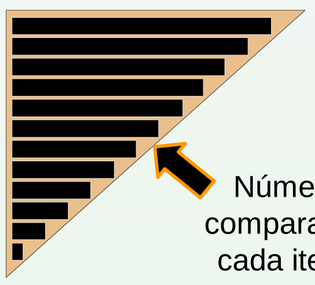
Placidus: A Platform for Formal Verification in Software Defined Networks

O **Placidus** é uma plataforma de verificação formal para redes definidas por software, e conta com dois módulos que verificam as seguintes propriedades: Conflitos e Redundâncias de regras lógicas formadas a partir de uma topologia, e *Reachability* dentro de uma rede.

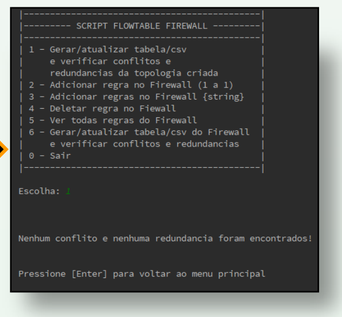
Conflitos e Redudâncias

Estrutura de Dados:
Lista com Tabelas *hash* (Chave: *Switch*, Valor: Lista de regras lógicas da rede)

Algoritmo:
É baseado na comparação de *matches* das regras, geradas a partir dos fluxos na rede, e suas ações por meio de sucessivas iterações, descartando uma regra a cada iteração para **reduzir o número de comparações pela metade**.



Número de comparações a cada iteração



… 00:04 ^ 00:02 ^ 0x0x806 -> *output* = 1

match ***ação***

Exemplo de regra formada

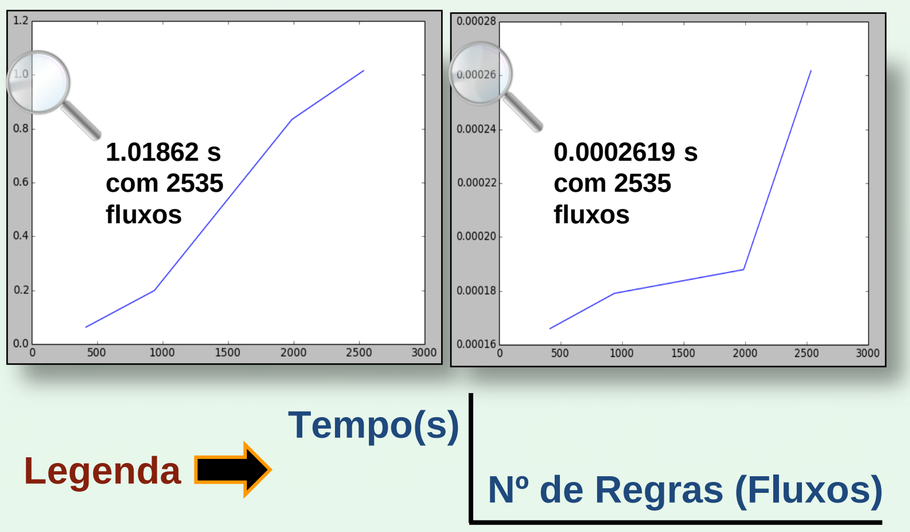
Reachability

Estrutura de Dados:
Lista com lista de predicados (regras lógicas).

Algoritmo:
A partir do arquivo *.csv* gerado pela coleta de dados, cada célula do arquivo, até então do tipo *string*, é transformada em objetos do tipo ***BitVector***. Essas células são agrupadas dentro de cada linha do *.csv* (regra). Então, a lista principal passa a conter *n* índices (predicados), formados por vetores com apenas **0** ou **1**.
Dentro desses vetores, os ***n-1*** *bits* mais significativos representam o *match*, e o *bit* mais significativo representa a ação da regra. O pacote de entrada é comparado via ***XNOR*** com os *n-1 bits* de cada índice da lista. Após o pacote ser encontrado, a operação ***AND*** é feita entre 1 e a ação da regra, se resultar em 1, o pacote passa, senão o pacote já é descartado.

Resultados Experimentais

Os experimentos foram rodados numa máquina Dell XPS 8700 com um processador Intel Core I7-4790 3.6GhZ de 8 núcleos e com 15.6 GB de memória RAM. Foram utilizadas diferentes topologias para se ter diferentes números de regras (fluxos) na rede.
Esses experimentos foram realizados com a utilização do emulador de rede, *Mininet*, e com o controlador *SDN*, *Floodlight OpenFlow Controller*.



Conclusão e Trabalhos Futuros

Nossa pesquisa na área de verificação formal de redes *SDN* tem o propósito de **tornar a rede mais robusta e confiável** em tempo real.

Trabalhos futuros incluem explorar outras propriedades globais de redes, como detecção de *loops* e *black holes* (verificar se existe sumidouros de pacotes na rede), e trabalhar no aprimoramento dos algoritmos já propostos para as aplicações desenvolvidas.

Salão UFRGS 2016 — Conhecimento Formação Inovação — paz no plural
Computer Networks UFRGS
INF Instituto de Informática UFRGS
UFRGS Universidade Federal do Rio Grande do Sul