Traffic-Aware Rule-Cache Assignment in SDN: Security Implications

S. Misra[1] N. Saha[1] R. Bhakta[2]

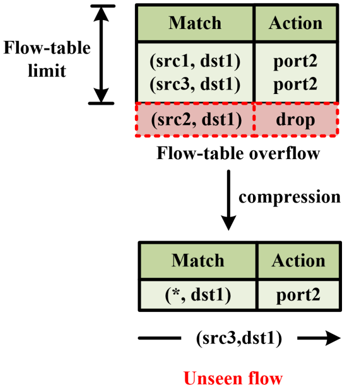
[1]Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

[2]Department of Computer Science and Engineering
National Institute of Technology, Durgapur

IEEE International Conference on Communications (ICC), 2020

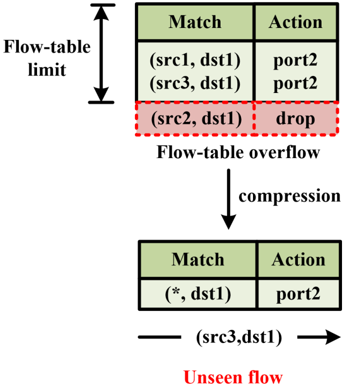
Motivation

- Match-action flow-rules for data-plane forwarding



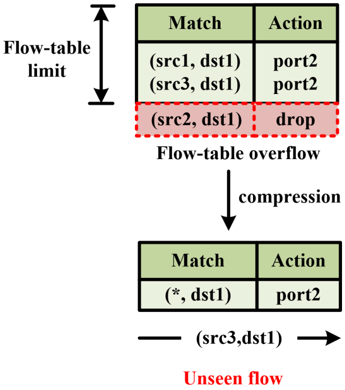
Motivation

- Match-action flow-rules for data-plane forwarding
- Limited flow-rule capacity at SDN switches



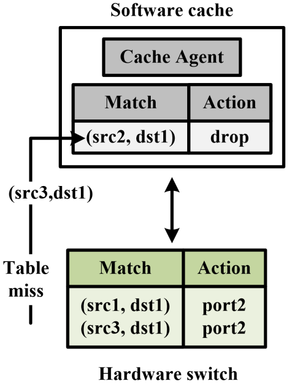
Motivation

- Match-action flow-rules for data-plane forwarding
- Limited flow-rule capacity at SDN switches
- Compression-based strategies → unseen flows



Motivation

- TCAM hardware augmented with inexpensive software switches[1] [2]

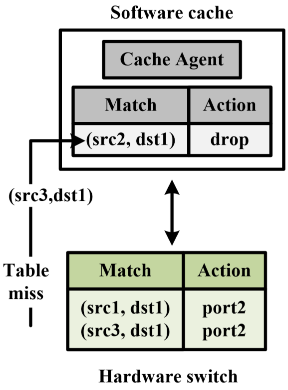


[1] N. Katta, O. Alipourfard, J. Rexford, and D. Walker, "CacheFlow: Dependency-Aware Rule-Caching for Software-Defined Networks," in *Proc. of the ACM SOSR*, New York, USA, 2016, pp. 6:1–6:12.

[2] A. Ruia, C. J. Casey, S. Saha, and A. Sprintson, "Flowcache: A cache-based approach for improving SDN scalability," in *Proc. of the IEEE INFOCOM Workshop*, April 2016, pp. 610–615

Motivation

- TCAM hardware augmented with inexpensive software switches[1] [2]
- Distributed software switches → scalability and fault tolerance

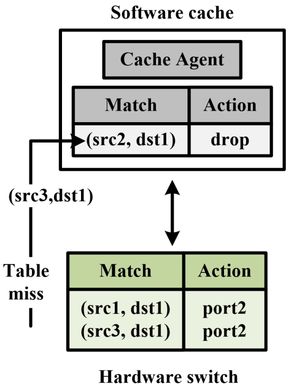


[1] N. Katta, O. Alipourfard, J. Rexford, and D. Walker, "CacheFlow: Dependency-Aware Rule-Caching for Software-Defined Networks," in *Proc. of the ACM SOSR*, New York, USA, 2016, pp. 6:1–6:12.

[2] A. Ruia, C. J. Casey, S. Saha, and A. Sprintson, "Flowcache: A cache-based approach for improving SDN scalability," in *Proc. of the IEEE INFOCOM Workshop*, April 2016, pp. 610–615

Motivation

- TCAM hardware augmented with inexpensive software switches[1] [2]
- Distributed software switches $\rightarrow$ scalability and fault tolerance
- Non-uniform latencies between hardware and software cache instances + many-to-many mapping $\rightarrow$ rule-cache assignment problem

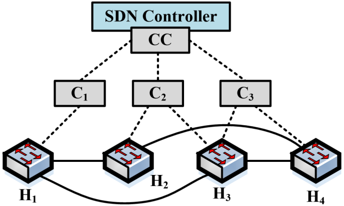


[1] N. Katta, O. Alipourfard, J. Rexford, and D. Walker, "CacheFlow: Dependency-Aware Rule-Caching for Software-Defined Networks," in *Proc. of the ACM SOSR*, New York, USA, 2016, pp. 6:1–6:12.

[2] A. Ruia, C. J. Casey, S. Saha, and A. Sprintson, "Flowcache: A cache-based approach for improving SDN scalability," in *Proc. of the IEEE INFOCOM Workshop*, April 2016, pp. 610–615

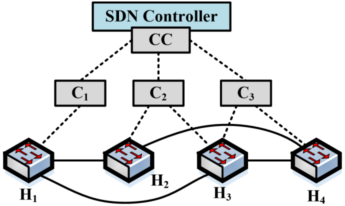
Rule-cache assignment

- Hardware switches $\mathcal{H}$ and software cache instances $\mathcal{C}$



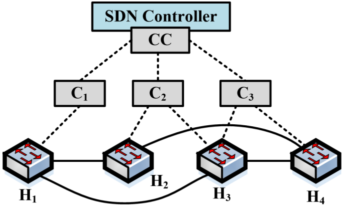
Rule-cache assignment

- Hardware switches $\mathcal{H}$ and software cache instances $\mathcal{C}$
- Scalability quota q_j^s of cache instance $j \in \mathcal{C}$



Rule-cache assignment

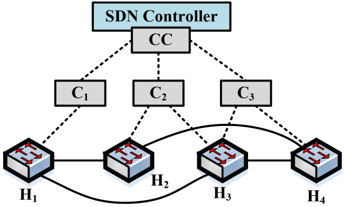
- Hardware switches $\mathcal{H}$ and software cache instances $\mathcal{C}$
- Scalability quota q_j^s of cache instance $j \in \mathcal{C}$
- Fault-tolerance quota q_i^{ft} of hardware switch $i \in \mathcal{H}$



Rule-cache assignment

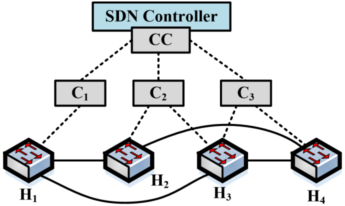
- Hardware switches $\mathcal{H}$ and software cache instances $\mathcal{C}$
- Scalability quota q_j^s of cache instance $j \in \mathcal{C}$
- Fault-tolerance quota q_i^{ft} of hardware switch $i \in \mathcal{H}$

- Minimize software cache instances (static)



Rule-cache assignment

- Hardware switches $\mathcal{H}$ and software cache instances $\mathcal{C}$
- Scalability quota q_j^s of cache instance $j \in \mathcal{C}$
- Fault-tolerance quota q_i^{ft} of hardware switch $i \in \mathcal{H}$



- Minimize software cache instances (static)
- Traffic-aware assignment (dynamic)

Rule-cache assignment: minimize cache instances

- Introduce binary variables $x(t)_{ij}$ denote assignment between switch i and cache j

Rule-cache assignment: minimize cache instances

- Introduce binary variables $x(t)_{ij}$ denote assignment between switch i and cache j
- Introduce binary variables $w(t)_j$ to keep track of cache instances

Rule-cache assignment: minimize cache instances

- Introduce binary variables $x(t)_{ij}$ denote assignment between switch i and cache j
- Introduce binary variables $w(t)_j$ to keep track of cache instances

$$\min \qquad \sum w(t)_j$$

Rule-cache assignment: minimize cache instances

- Introduce binary variables $x(t)_{ij}$ denote assignment between switch i and cache j
- Introduce binary variables $w(t)_j$ to keep track of cache instances

$$\begin{aligned} \min \quad & \sum w(t)_j \\ \text{subject to} \quad & \sum_{j \in \mathcal{C}} x(t)_{ij} = q_i^{ft}, \quad \forall i \in \mathcal{H} \end{aligned}$$

Rule-cache assignment: minimize cache instances

- Introduce binary variables $x(t)_{ij}$ denote assignment between switch i and cache j
- Introduce binary variables $w(t)_j$ to keep track of cache instances

$$
\begin{aligned}
\text{min} \quad & \sum w(t)_j \\
\text{subject to} \quad & \sum_{j \in \mathcal{C}} x(t)_{ij} = q_i^{ft}, \quad \forall i \in \mathcal{H} \\
& \sum_{i \in \mathcal{H}} x(t)_{ij} \leq q_j^s w(t)_j, \quad \forall j \in \mathcal{C}
\end{aligned}
$$

Rule-cache assignment: minimize cache instances

- Introduce binary variables $x(t)_{ij}$ denote assignment between switch i and cache j
- Introduce binary variables $w(t)_j$ to keep track of cache instances

$$
\begin{aligned}
\text{min} \quad & \sum w(t)_j \\
\text{subject to} \quad & \sum_{j \in \mathcal{C}} x(t)_{ij} = q_i^{ft}, \quad \forall i \in \mathcal{H} \\
& \sum_{i \in \mathcal{H}} x(t)_{ij} \leq q_j^{s} w(t)_j, \quad \forall j \in \mathcal{C} \\
& x(t)_{ij} \leq w(t)_j, \quad \forall i \in \mathcal{H}, \forall j \in \mathcal{C}
\end{aligned}
$$

Rule-cache assignment: traffic-aware assignment

- Delay between switch i and software cache j

Rule-cache assignment: traffic-aware assignment

- Delay between switch i and software cache j
- Control overhead in timeslot t

Rule-cache assignment: traffic-aware assignment

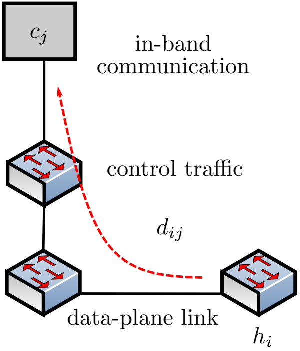
- Delay between switch i and software cache j
- Control overhead in timeslot t

- Delay $\delta(t) \rightarrow$ propagation delay + queuing delay

Rule-cache assignment: traffic-aware assignment

- Delay between switch i and software cache j
- Control overhead in timeslot t

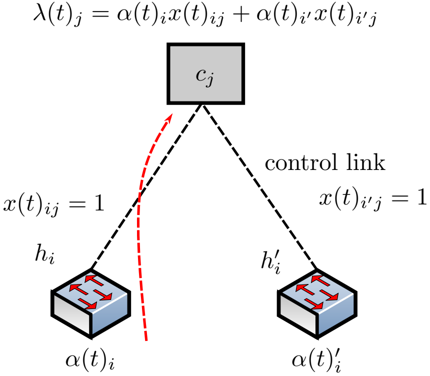
- Delay $\delta(t) \rightarrow$ propagation delay + queuing delay
- Propagation delay $\rightarrow \sum_i \sum_j \frac{d_{ij}}{v}$



Rule-cache assignment: traffic-aware assignment

- Delay between switch i and software cache j
- Control overhead in timeslot t

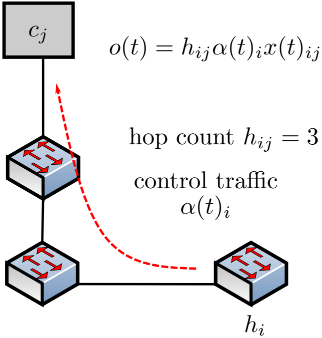
- Delay $\delta(t) \rightarrow$ propagation delay + queuing delay
- Propagation delay $\rightarrow \sum_i \sum_j \frac{d_{ij}}{v}$
- Queuing delay $\rightarrow \sum_j \frac{1}{\mu_j - \lambda(t)_j}$



Rule-cache assignment: traffic-aware assignment

- Delay between switch i and software cache j
- Control overhead in timeslot t

- Delay $\delta(t) \rightarrow$ propagation delay + queuing delay
- Propagation delay $\rightarrow \sum_i \sum_j \frac{d_{ij}}{v}$
- Queuing delay $\rightarrow \sum_j \frac{1}{\mu_j - \lambda(t)_j}$
- Control overhead $o(t) = \sum_j \sum_j h_{ij}\alpha(t)_i x(t)_{ij}$



Rule-cache assignment: traffic-aware assignment

The problem can be formulated as an integer program as follows:

Rule-cache assignment: traffic-aware assignment

The problem can be formulated as an integer program as follows:

$$\min \qquad \eta\delta(t) + (1-\eta)o(t)$$

Rule-cache assignment: traffic-aware assignment

The problem can be formulated as an integer program as follows:

$$\begin{aligned} \text{min} \quad & \eta\delta(t) + (1-\eta)o(t) \\ \text{subject to} \quad & \lambda(t)_j \leq \beta\mu_j \quad \forall j \in \mathcal{C} \end{aligned}$$

Rule-cache assignment: traffic-aware assignment

The problem can be formulated as an integer program as follows:

$$
\begin{aligned}
\text{min} \quad & \eta\delta(t) + (1-\eta)o(t) \\
\text{subject to} \quad & \lambda(t)_j \leq \beta\mu_j \quad \forall j \in \mathcal{C} \\
& x(t)_{ij} \in \{0,1\}, \forall i, j \\
& \sum_{j \in \mathcal{C}} x(t)_{ij} = q_i^{ft}, \quad \forall i \in \mathcal{H} \\
& \sum_{i \in \mathcal{H}} x(t)_{ij} \leq q_j^{s}, \quad \forall j \in \mathcal{C}
\end{aligned}
$$

Rule-cache assignment: traffic-aware assignment

The problem can be formulated as an integer program as follows:

$$
\begin{aligned}
\min \quad & \eta\delta(t) + (1-\eta)o(t) \\
\text{subject to} \quad & \lambda(t)_j \leq \beta\mu_j \quad \forall j \in \mathcal{C} \\
& x(t)_{ij} \in \{0,1\}, \forall i, j \\
& \sum_{j \in \mathcal{C}} x(t)_{ij} = q_i^{ft}, \quad \forall i \in \mathcal{H} \\
& \sum_{i \in \mathcal{H}} x(t)_{ij} \leq q_j^{s}, \quad \forall j \in \mathcal{C}
\end{aligned}
$$

- Time consuming to solve for large instances!

Rule-cache assignment: traffic-aware assignment

The problem can be formulated as an integer program as follows:

$$
\begin{aligned}
\text{min} \quad & \eta\delta(t) + (1-\eta)o(t) \\
\text{subject to} \quad & \lambda(t)_j \leq \beta\mu_j \quad \forall j \in \mathcal{C} \\
& x(t)_{ij} \in \{0,1\}, \forall i,j \\
& \sum_{j \in \mathcal{C}} x(t)_{ij} = q_i^{ft}, \quad \forall i \in \mathcal{H} \\
& \sum_{i \in \mathcal{H}} x(t)_{ij} \leq q_j^{s}, \quad \forall j \in \mathcal{C}
\end{aligned}
$$

- Time consuming to solve for large instances!
- Efficient solution needed for dynamic system

Our Approach

- Based on matching theory concepts[1]

[1] A. Roth and M. Sotomayor, *Two-Sided Matching: A Study in Game Theoretic Modeling and Analysis*. Cambridge University Press, 1992

Our Approach

- Based on matching theory concepts[1]
- First stage: network statistics collection

[1] A. Roth and M. Sotomayor, *Two-Sided Matching: A Study in Game Theoretic Modeling and Analysis*. Cambridge University Press, 1992

Our Approach

- Based on matching theory concepts[1]
- First stage: network statistics collection
- Second stage: build preference relations

[1] A. Roth and M. Sotomayor, *Two-Sided Matching: A Study in Game Theoretic Modeling and Analysis*. Cambridge University Press, 1992

Our Approach

- Based on matching theory concepts[1]
- First stage: network statistics collection
- Second stage: build preference relations
  - Hardware switches' objective → minimize delay $\delta(t)$

[1] A. Roth and M. Sotomayor, *Two-Sided Matching: A Study in Game Theoretic Modeling and Analysis*. Cambridge University Press, 1992

Our Approach

- Based on matching theory concepts[1]
- First stage: network statistics collection
- Second stage: build preference relations
  - Hardware switches' objective $\rightarrow$ minimize delay $\delta(t)$
  - Software caches' objective $\rightarrow$ minimize control overhead $o(t)$

[1] A. Roth and M. Sotomayor, *Two-Sided Matching: A Study in Game Theoretic Modeling and Analysis*. Cambridge University Press, 1992

Our Approach

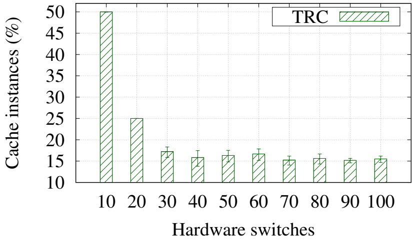
- Based on matching theory concepts[1]
- First stage: network statistics collection
- Second stage: build preference relations
  - Hardware switches' objective → minimize delay $\delta(t)$
  - Software caches' objective → minimize control overhead $o(t)$
- Third stage: two sided matching

1: Each $j \in \mathcal{C}$ sends association request to its preferred subset $C_i(\mathcal{H})$.
2: Each $i \in \mathcal{H}$ refuses all expect the preferred q_i^{ft} cache instances.
3: **repeat**
4: Each $j \in \mathcal{C}$ sends association request to its preferred subset $C_i(\mathcal{H})$, including those already sent to who have not refused it yet.
5: Each $i \in \mathcal{H}$ refuses all except the preferred q_i^{ft} cache instances.
6: **until** convergence to a pairwise-stable outcome

[1] A. Roth and M. Sotomayor, *Two-Sided Matching: A Study in Game Theoretic Modeling and Analysis*. Cambridge University Press, 1992

Results

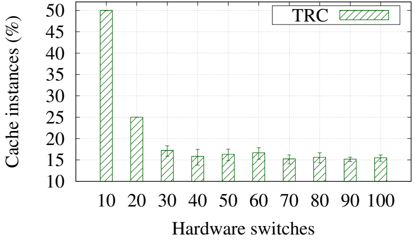
- Solved minimum cache assignment using GLPK solver



[1] A. Ruia, C. J. Casey, S. Saha, and A. Sprintson, "Flowcache: A cache-based approach for improving SDN scalability," in *Proc. of the IEEE INFOCOM Workshop*, April 2016, pp. 610–615

Results

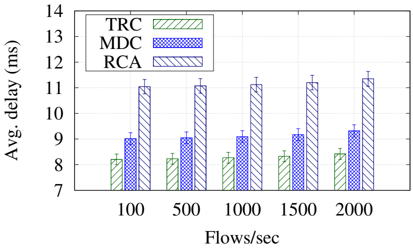
- Solved minimum cache assignment using GLPK solver
- No. of caches → approx. 15% of hardware switches



[1] A. Ruia, C. J. Casey, S. Saha, and A. Sprintson, "Flowcache: A cache-based approach for improving SDN scalability," in *Proc. of the IEEE INFOCOM Workshop*, April 2016, pp. 610–615

Results

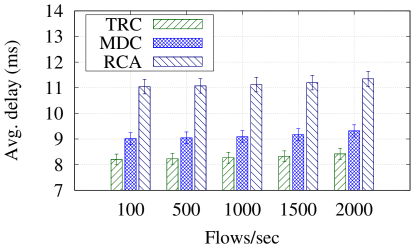
- Solved minimum cache assignment using GLPK solver
- No. of caches $\rightarrow$ approx. 15% of hardware switches
- Benchmarks: random assignment (RCA) and minimum distance assignment (MDC)[1]



[1] A. Ruia, C. J. Casey, S. Saha, and A. Sprintson, "Flowcache: A cache-based approach for improving SDN scalability," in *Proc. of the IEEE INFOCOM Workshop*, April 2016, pp. 610–615

Results

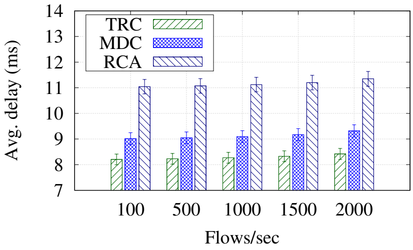
- Solved minimum cache assignment using GLPK solver
- No. of caches → approx. 15% of hardware switches
- Benchmarks: random assignment (RCA) and minimum distance assignment (MDC)[1]
- Proposed scheme (TRC) reduces delay by approx. 10% and 35% compared to MDC and RCA



[1] A. Ruia, C. J. Casey, S. Saha, and A. Sprintson, "Flowcache: A cache-based approach for improving SDN scalability," in *Proc. of the IEEE INFOCOM Workshop*, April 2016, pp. 610–615

Results

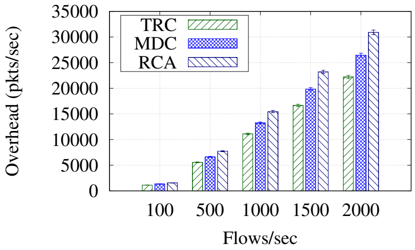
- Solved minimum cache assignment using GLPK solver
- No. of caches → approx. 15% of hardware switches
- Benchmarks: random assignment (RCA) and minimum distance assignment (MDC)[1]
- Proposed scheme (TRC) reduces delay by approx. 10% and 35% compared to MDC and RCA
- MDC suffers due to load imbalance



[1] A. Ruia, C. J. Casey, S. Saha, and A. Sprintson, "Flowcache: A cache-based approach for improving SDN scalability," in *Proc. of the IEEE INFOCOM Workshop*, April 2016, pp. 610–615

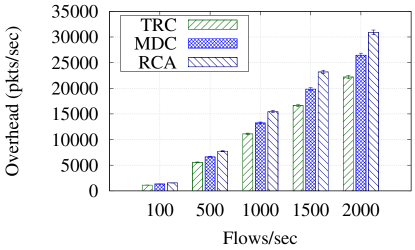
Results

- Proposed scheme (TRC) reduces overhead by approx. 19% and 39% compared to MDC and RCA



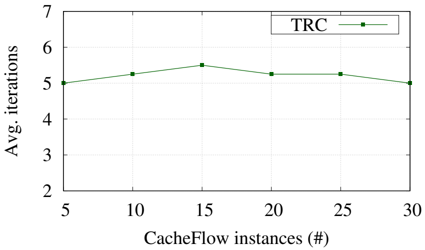
Results

- Proposed scheme (TRC) reduces overhead by approx. 19% and 39% compared to MDC and RCA
- Effect of traffic rate significant



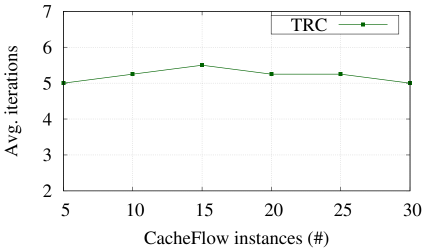
Results

- Proposed scheme (TRC) reduces overhead by approx. 19% and 39% compared to MDC and RCA
- Effect of traffic rate significant
- Algorithm quickly converges in a few iterations



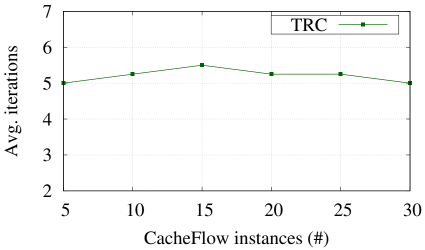
Results

- Proposed scheme (TRC) reduces overhead by approx. 19% and 39% compared to MDC and RCA
- Effect of traffic rate significant
- Algorithm quickly converges in a few iterations
- No. of cache instances does not affect the iterations significantly



Results

- Proposed scheme (TRC) reduces overhead by approx. 19% and 39% compared to MDC and RCA
- Effect of traffic rate significant
- Algorithm quickly converges in a few iterations
- No. of cache instances does not affect the iterations significantly
- Efficient for dynamic re-assignment with varying traffic conditions



Security Implications

- Attacks specific to SDN include denial-of-service by attacking control-plane or flow-table[1]

[1] R. Kandoi and M. Antikainen, "Denial-of-service attacks in openflow sdn networks," in *Proc. of the IFIP/IEEE International Symposium on Integrated Network Management*, May 2015, pp. 1322–1326.

Security Implications

- Attacks specific to SDN include denial-of-service by attacking control-plane or flow-table[1]
- Control plane attack $\rightarrow$ intentionally crafted packets to trigger table-miss

[1] R. Kandoi and M. Antikainen, "Denial-of-service attacks in openflow sdn networks," in *Proc. of the IFIP/IEEE International Symposium on Integrated Network Management*, May 2015, pp. 1322–1326.

Security Implications

- Attacks specific to SDN include denial-of-service by attacking control-plane or flow-table[1]
- Control plane attack $\rightarrow$ intentionally crafted packets to trigger table-miss
- Flow-table attack $\rightarrow$ design packets to install large no. of flow-rules

[1] R. Kandoi and M. Antikainen, "Denial-of-service attacks in openflow sdn networks," in *Proc. of the IFIP/IEEE International Symposium on Integrated Network Management*, May 2015, pp. 1322–1326.

Security Implications

- Attacks specific to SDN include denial-of-service by attacking control-plane or flow-table[1]
- Control plane attack $\rightarrow$ intentionally crafted packets to trigger table-miss
- Flow-table attack $\rightarrow$ design packets to install large no. of flow-rules

- Software cache architecture provides large flow-space; reduces probability of table-miss and flow-table overflow

[1] R. Kandoi and M. Antikainen, "Denial-of-service attacks in openflow sdn networks," in *Proc. of the IFIP/IEEE International Symposium on Integrated Network Management*, May 2015, pp. 1322–1326.

Security Implications

- Attacks specific to SDN include denial-of-service by attacking control-plane or flow-table[1]
- Control plane attack $\rightarrow$ intentionally crafted packets to trigger table-miss
- Flow-table attack $\rightarrow$ design packets to install large no. of flow-rules

- Software cache architecture provides large flow-space; reduces probability of table-miss and flow-table overflow
- Reduces chances of unseen flows compared to compression-based strategies

[1] R. Kandoi and M. Antikainen, "Denial-of-service attacks in openflow sdn networks," in *Proc. of the IFIP/IEEE International Symposium on Integrated Network Management*, May 2015, pp. 1322–1326.

Security Implications

- Attacks specific to SDN include denial-of-service by attacking control-plane or flow-table[1]
- Control plane attack $\rightarrow$ intentionally crafted packets to trigger table-miss
- Flow-table attack $\rightarrow$ design packets to install large no. of flow-rules

- Software cache architecture provides large flow-space; reduces probability of table-miss and flow-table overflow
- Reduces chances of unseen flows compared to compression-based strategies
- Dynamic re-assignment capability reduces network overhead

[1] R. Kandoi and M. Antikainen, "Denial-of-service attacks in openflow sdn networks," in *Proc. of the IFIP/IEEE International Symposium on Integrated Network Management*, May 2015, pp. 1322–1326.

Thank You