ApproSync

Approximate State Synchronization for Programmable Networks

Xiang Chen, Qun Huang, Dong Zhang, Haifeng Zhou, Chunming Wu
Control Plane (CP)
Applications

Policies
States

Data Plane (DP) Programmable Switches

Background | Problems | Challenges | Design | Evaluation | Summary
State: Historical Packet Processing Information

e.g., Count-Min Sketch running on a Tofino switch

State = Set of counter values; A state value = A counter value
State Sync: Making States in CP and DP Consistent

1. Bottom-Up Sync.
   - State Read
   - Read (DP → CP)

Control Plane (CP) Applications

Data Plane (DP) Programmable Switches

Data Plane States (in switch ASICs)

Background | Problems | Challenges | Design | Evaluation | Summary
State Sync: Making States in CP and DP Consistent

Control Plane (CP) Applications

1. Bottom-Up Sync.
   - State Read
   - Read (DP → CP)

2. Top-Down Sync.
   - State Write
   - Write (CP → DP)

Policies

Data Plane States (in switch ASICs)

Data Plane (DP) Programmable Switches

Background | Problems | Challenges | Design | Evaluation | Summary
Requirements

1. **Low latency** for latency-sensitive apps (e.g., Anomaly Detect) complete state sync within a small time

2. **High accuracy** for apps to make correct decisions minimize state divergence (i.e., difference) between CP and DP
Limitations of Existing Solutions (Switch OS)

High Latency in Switch OS

Sync state values via PCIe and TCP

PCIe and TCP bandwidth < 100 Gbps

Transfer all state updates

High resource consumption >> 100 Gbps
Limitations of Existing Solutions (Switch OS)

Collect $2^{16}$ counter values via OS of a Tofino switch

Our benchmark:

$>10s$ latency
Limitations of Existing Solutions (Traffic Mirroring)

State Loss in Traffic Mirroring

Mirror state values to CP

Low latency via bypassing switch OS

State Loss due to limited link capacity
Limitations of Existing Solutions (Traffic Mirroring)

Collect $2^{16}$ state values under 40-120 Gbps input traffic rate

Our benchmark: up to 60% State Loss

(Use a 40 Gbps link for state transfer)
Impact on Applications (Heavy Hitter Detection)

Collect a hash table with $2^{16}$ entries from a Tofino switch.

High Latency and State Loss seriously affects App accuracy.

(a) Impact of High Latency

(b) Impact of State Loss
Can we achieve both **Low Latency** and **High Accuracy**?

**Low Latency**: OS bypassing

Sync states between switch ASICs and CP (w/o invoking OS)
Can we achieve both **Low Latency** and **High Accuracy**?

**Low Latency**: OS bypassing

Sync states between switch ASICs and CP (*w/o invoking OS*)

**High Accuracy**

- **State loss** due to limited link capacity (tens of Gbps)
- **Switch limitations** (e.g., <10 MB memory)
- **Challenge**: How to handle state loss under limitations?
Observation

Applications often tolerate a small state divergence (e.g., <1%).

For the heavy hitter, UDP flood, and superspreader detection:

1. DP value $v_1 = 100$; CP value $v_2 = 99$; div rate = $|v_1-v_2|/v_1 \times 100\% = 1\%$.
Observation

Applications often tolerate a small state divergence (e.g., <1%).

For heavy hitter, UDP flood, and superspreader detection:

State divergence < 1% → App-level error < 2%
ApproSync — Approximate State Sync

1. Bypass switch OS → Low Latency

2. Allow a small divergence (err) → Low Resource Consumption
    → No State Loss → High Accuracy

- high latency
  - full accuracy
- low latency
  - high accuracy
- low latency
  - low accuracy

switch OS  ApproSync  traffic mirroring

Background | Problems | Challenges | Design | Evaluation | Summary
**ApproSync — Approximate State Sync**

**Design#1: Hash Table in Switch ASIC**

1. *Aggregate state updates* with same locations

   ![Diagram of Switch ASIC with packets A and B and hash table updates](image)

   - **Update#1**: \((1,1), 1\) - Change value in \((1,1)\) to 1
   - **Update#2**: \((1,1), 2\) - Change value in \((1,1)\) to 2
ApproSync — Approximate State Sync

Design#1: Hash Table in Switch ASIC

1. Aggregate state updates with same locations

Update#1: \(((1,1), 1)\)
Update#2: \(((1,1), 2)\)

If send all updates
link saturation, state loss
ApproSync — Approximate State Sync

Design#1: Hash Table in Switch ASIC

1. Aggregate state updates with same locations

Packet A

Packet B

Switch ASIC

Update#1: \((1,1), 1\)

Update#2: \((1,1), 2\)

Aggregation by Hash Table

Aggregated Update: \((1,1), 2\)

If send all updates

link saturation, state loss

Send to CP
**ApproSync — Approximate State Sync**

Design#1: Hash Table in Switch ASIC

1. Aggregate state updates with same locations
2. Bound state divergence between DP and CP

DP value: $v_1$  
CP value: $v_2$  
State divergence: $\text{div} = |v_1 - v_2|$

Bound $\text{div} = |v_1 - v_2| \leq \text{threshold t}$
**Example of Hash Table** (threshold $t=1$)

**Hash Table $H$**

<table>
<thead>
<tr>
<th>Loc</th>
<th>Val</th>
<th>Old</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Loc:** Counter ID  
**Val:** Latest state value in DP  
**Old:** Last state value sent to CP (i.e., value in CP)

**Switch ASIC**

Value[1] = 0  
Value[2] = 0

**Controller**

Value[1] = 0  
Value[2] = 0
Example of Hash Table (threshold t=1)

Hash Table $H$

<table>
<thead>
<tr>
<th>Loc</th>
<th>Val</th>
<th>Old</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Loc: Counter ID
Val: Latest state value in DP
Old: Last state value sent to CP (i.e., value in CP)

Update $H[1].value = 1$

Switch ASIC
Value[1] = 1
Value[2] = 0

Controller
Value[1] = 0
Value[2] = 0
Example of Hash Table (threshold t=1)

<table>
<thead>
<tr>
<th>Loc</th>
<th>Val</th>
<th>Old</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Loc: Counter ID
Val: Latest state value in DP
Old: Last state value sent to CP (i.e., value in CP)

State divergence (div) = |Val - Old| = 1 - 0 = 1 ≤ t

No need to sync since div is small

(1, 1)

Switch ASIC
Value[1] = 1
Value[2] = 0

Controller
Value[1] = 0
Value[2] = 0

(div refers to state divergence)
Example of Hash Table (threshold \(t=1\))

<table>
<thead>
<tr>
<th>Loc</th>
<th>Val</th>
<th>Old</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Loc: Counter ID
Val: Latest state value in DP
Old: Last state value sent to CP (i.e., value in CP)

\(H[1].value = 2\): Aggregate with previous update

Switch ASIC

Value[1] = 2
Value[2] = 0

Controller

Value[1] = 0
Value[2] = 0
**Example of Hash Table (threshold t=1)**

<table>
<thead>
<tr>
<th>Loc</th>
<th>Val</th>
<th>Old</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Loc: Counter ID  
Val: Latest state value in DP  
Old: Last state value sent to CP (i.e., value in CP)

\[ \text{div} = \text{Val} - \text{Old} = 2 - 0 = 2 > t \]

Sync H[1] since div is large!

Switch ASIC  
Value[1] = 2  
Value[2] = 0

Controller  
Value[1] = 2  
Value[2] = 0

(div refers to state divergence)
Example of Hash Table (threshold t=1)

<table>
<thead>
<tr>
<th>Hash Table H</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loc</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

Takeaway#1:
- w/o Hash Table: sync all state updates
- w/ Hash Table: sync one aggregated update
  - reduce link load by 50%

Hash Table can reduce link load

Takeaway#2:
- State divergence (div) \( \leq \) threshold \( t = 1 \)
ApproSync — Approximate State Sync

Design#1: Hash Table in Switch ASIC

1. Aggregate state updates with same locations
2. Allow a small state divergence to reduce link load

Design#2: Rate Control in Switch ASIC

Adaptively tune threshold \( t \) w.r.t. incoming traffic rate

Design#3: Reliable and Atomic State Write

Please refer to our paper :-)}
Implementation

ApproSync is written in P4 language and runs on Tofino switches

Support State Read and State Write

Protocol for State Transfer

Workflow of Switch ASIC
Evaluation

Testbed: Barefoot Tofino Switches + Commodity Servers

Workload: CAIDA 2018 trace, 16 stateful P4 applications

Comparison: Switch OS, Traffic Mirroring, *Flow (ATC’18)

(1) Can ApproSync achieve low latency and high accuracy?
(2) Can ApproSync bring benefits to real applications?
Evaluation

Low-Latency State Synchronization

Order-of-Magnitude Latency Reduction
Evaluation

Accurate State Synchronization

w/o ApproSync’s Hash Table

w/ Hash Table: Zero State Loss

Threshold $t$ of Hash Table

AS-Dyn = Original ApproSync

0% State Loss even w/ 200 Gbps
Evaluation

Performance of state r/w in 16 stateful P4 applications

Read

Low-Latency State Sync for 16 Applications

Write
Evaluation

Accuracy of Collecting $2^{16}$ Values (e.g., Count-Min Sketch)

Accurate State Sync (close to ideal situation)
Takeaways

Existing State Sync: **High Latency** or **Low Accuracy**

Challenge: handle **State Loss** under **switch limitations**

Observation: Apps tolerate a **small** state divergence

ApproSync: **Approximate** State Sync

(1) OS bypassing for **low latency** (2) Hash table for **high accuracy**
Thank you very much!

Xiang Chen, Qun Huang, Dong Zhang, Haifeng Zhou, Chunming Wu

Email: wasdnsxchen@gmail.com    Page: wasdns.github.io
Backup Slides
State Loss Example
1. State Loss → High State Divergence

State Updates:

Switch ASIC
Value[1] = 2
Value[2] = 1

Controller
Value[1] = 0
Value[2] = 0

state location new value = 1

link (≤ 2 values)
1. State Loss → High State Divergence

State Updates

Switch ASIC
Value[1] = 2
Value[2] = 1

Controller
Value[1] = 0
Value[2] = 0

state location
new value=1

link (≤ 2 values)
1. State Loss $\rightarrow$ High State Divergence

State Updates:

- Switch ASIC
  - Value[1] = 2
  - Value[2] = 1

- Controller
  - Value[1] = 1
  - Value[2] = 1

Loss link ($\leq 2$ values)
2. Limitations of Switch ASIC

Memory Limitation

at most 10 MB RAM memory

Computation Limitation

a few memory accesses; forbid complex operations (e.g., loop)

Existing methods (e.g., retransmission) are not deployable
Rate Control
Rate Control

Traffic mirroring push *every* state update to CP:

Emitted rate $R = T$ (incoming traffic rate) $\rightarrow$ State Loss
Rate Control

Traffic mirroring push *every* state update to CP:

Emitted rate $R = T$ (incoming traffic rate) $\rightarrow$ State Loss

ApproSync uses *Hash Table* (threshold $t$):

**Bound** state divergence: $\text{div} \leq t$

If $\text{div} > t$, DP state update is sync to CP

Send a update every $t$ updates: $R \approx \lceil T/t \rceil$
Rate Control

Emitted rate $R \approx \lceil T/t \rceil$

Link capacity (# state updates / second) $M$

To avoid state loss: $R \leq M$

$R \approx \lceil T/t \rceil \leq M \rightarrow t \geq \lceil T/M \rceil$

ApproSync tunes $t = \lceil T/M \rceil$

Achieve minimal state divergence w/o state loss

*please refer to our paper for more details*
Example of Rate Control

Traffic rate $T = 10^7$ updates/s

Switch ASIC

Link capacity $M = 7.8 \times 10^7$ updates/s

$10^7 < 7.8 \times 10^7$

Threshold $t = 1$ (sync every update) is sufficient

Link will not be saturated, so no state loss occurs
Example of Rate Control

Traffic rate $T_{10^8 \text{ updates/s}}$ -> Switch ASIC -> Link capacity $M_{7.8 \times 10^7 \text{ updates/s}}$

$10^8 > 7.8 \times 10^7$
Example of Rate Control

Traffic rate $T = 10^8$ updates/s

Link capacity $M = 7.8 \times 10^7$ updates/s

$10^8 > 7.8 \times 10^7$

Tune $t = 2$ (sync 1 update every 2 updates)

$10^8 > 7.8 \times 10^7 \rightarrow 10^8/t < 7.8 \times 10^7$ ($t=2$)

Avoid link overload and state loss
More Results
Evaluation

Low-Latency State Read and State Write

Order-of-Magnitude Latency Reduction for State Write