State of the art and challenges on consistency management at switch and controller layers

Rémi Oudin, LIP6

April 10, 2018
State of the art and challenges on consistency management at switch and controller layers

Introduction

In the SDN paradigm, inconsistencies can appear both on the control plane and on the data plane\(^1\).

**State consistency:** Distributed state across cluster members is replicated. Requires every controller to have the same global view.

**Version update consistency:** Multiple controllers have the newest state rather than hold the old state of the network.

**Rules update consistency:** Controllers and switches need to keep the same forwarding policies for stable forwarding.

---

\(^1\) Zhang et al. “A survey on software defined networking with multiple controllers” in *Journal of Network and Computer Applications*, 2018.
Rules update consistency: Data plane

Consistent Network Update

Given a consistency property to preserve during a network update, what solutions exist, with which guarantees?

General statement

Given a set of connected devices, with routing rules installed on them, and given a network update, which is a state to be reached (addition, deletion and modifications of flows) find a sequence of operations that preserve, if possible, a consistency property. This set of operations should optimize a performance criteria, and may have some final operational sequences.
State of the art and challenges on consistency management at switch and controller layers

Rules update consistency: Data plane
State of the art and challenges on consistency management at switch and controller layers

Rules update consistency: Data plane
What if we add first a rule from (3) to (4)?
State of the art and challenges on consistency management at switch and controller layers

Rules update consistency: Data plane
Different consistency properties

**Connectivity:** broadly speaking blackhole and loop freedom. Extremely basic properties, yet absolutely necessary.

**Policy:** Enforcing a policy, like «Per packet consistency», «Waypoint Enforcement», «Per flow consistency». Necessary for example to enforce packets to go through a firewall.

**Capacity:** If possible, the network update should be congestion free. Note that it is not always possible. Moreover, it strongly depends on buffer sizes, etc. The property here is to avoid ongoing bandwidth violation of any node.
Operations considered

**Rule replacement:** Compute an order in which initial rules are replaced by the corresponding final rules.

**Rule addition:** Use helper rules to guarantee consistency during the update.
Performance goals

**Link-based:** Focus on aiming to make links available as soon as possible.

**Round-based:** Minimize the total makespan by computing a schedule of rounds of updates that can be done simultaneously.

**Cross-Flow:** in presence of multiple flows, minimize the number of interactions with the switch, or minimize the congestion.
Two definitions of loop-freedom are possible\(^2\):

**Strong Loop-Freedom** At any point of time, the forwarding rules store at the switches should be loop-free.

**Relaxed Loop-Freedom** Forwarding rules along the path from a source to a destination are loop-free: only a small number of old packets may temporarily be forwarded along loops.

List of results

Many different results, according to the required consistency.

**Strong Loop-Freedom:** NP-Hard for round-based performance if number
of rounds is greater than 3. If link-based, NP-hard.

**Relaxed Loop-Freedom:** $O(\log n)$-round update always exists. NP-hard
to decide if $x$ nodes can be updated in a LF manner.

**Per-packet consistency:** 2-phase commit$^3$, restricted 2P-commit$^4$,
per-switch update protocol$^5$.

**Waypoint-Enforcement** WayUp (does not guarantee connectivity, but
polynomial) or Mixed Integer Programming$^6$ (exponential).

---


$^4$ Vissicchio et al. “Safe Update of Hybrid SDN Networks” in *IEEE/ACM

’13*, 2013.

$^6$ Ludwig et al. “Good Network Updates for Bad Packets: Waypoint Enforcement
List of results

Capacity-aware consistency is more complicated. There are different models, if the flows are splittable or not, if it allows intermediate paths or not...

**zUpdate**\(^7\) requires some slack on the links. Achieves in polynomial time.

**MCUP**\(^8\) polynomial for update without intermediate paths. No bound on the number of updates. Approximation algorithms exist.

**2PC**\(^9\) No bandwidth guarantee, but fixed number of updates.

---


\(^8\) Zheng et al. “Minimizing Transient Congestion during Network Update in Data Centers” in, 2015.

From Theory to Practice

There are some practical challenges in order to ensure consistent network updates:

- Ensuring operations are applied in hardware
- Working around device limitations (delay when adding a rule, limitations of statistics request)
- Avoiding conflict between multiple control-planes.
- Updating the control plane.
- Dealing with events occurring during an update.
State of the art and challenges on consistency management at switch and controller layers

On the control plane

Consistency between distributed controllers

The problem here is to keep a consistent state in a set of controllers. This leverages multiple problematics:\(^{10}\):

- Physically distributed or centralised?
- Logically centralised or distributed?
- Flat or hierarchical structure?
- Static or dynamic allocation of the switches?
- What consistency should be enforced?

---

\(^{10}\) Blial, Ben Mamoun, and Redouane. “An Overview on SDN Architectures with Multiple Controllers” in, 2016.
State of the art and challenges on consistency management at switch and controller layers

- On the control plane
- State consistency

**Figure:** A multiple controller SDN network...
State of the art and challenges on consistency management at switch and controller layers

- On the control plane
- State consistency

Figure: ... with a control plane link down
State of the art and challenges on consistency management at switch and controller layers

On the control plane

State consistency

Figure: ...with a controller down
State of the art and challenges on consistency management at switch and controller layers

- On the control plane
- State consistency

Figure: ...with a switch down
State of the art and challenges on consistency management at switch and controller layers

- On the control plane
- State consistency

Figure: ... with a data plane link down
State of the art and challenges on consistency management at switch and controller layers

On the control plane

State consistency

Consistencies for state consistency

**Strong consistency**\(^{11}\): Slow, CPU intensive. Sync between each operation.

- Exists in ONOS: Raft algorithm

**Eventual consistency**: Fast, but reliable mostly if few writes are done. Guarantees that if no new updates are made, all accesses will eventually return the last updated value.

- Exists in ONOS: State machine in ONOS, with "anti-entropy" process.

**Adaptive consistency**\(^{12}\): Consistency level is adapted according to the load (read and writes). It can go from strong down to eventual consistency.

«State synchronisation occurs according to performance and consistency constraints set by the application at runtime.»

---


State of the art and challenges on consistency management at switch and controller layers

- On the control plane
- State consistency

Examples: Strong consistency
Examples: Strong consistency
State of the art and challenges on consistency management at switch and controller layers

- On the control plane
  - State consistency

Examples: Strong consistency

Synchronisation
Examples: Strong consistency
Examples: Eventual Consistency

Figure: At some point, a switch goes down, or a link is broken.
Examples: Eventual Consistency

Figure: The controller detects the problem.
Examples: Eventual Consistency

Figure: The other controller is *eventually* notified after a repair protocol. (Ex: Gossip based protocol, fix-on-read, fix-on-write)
Examples: Eventual Consistency

Figure: The controller is notified from the change, state consistency is restored.
Examples: Adaptive consistency$^{13}$

How it works?

Each controller is given a number of credits. When all credits are consumed, a synchronisation happens. The consistency level defines the maximum *non-synchronisation* time allowed in the system.

---

State of the art and challenges on consistency management at switch and controller layers

- On the control plane
- State consistency

Examples: Adaptive consistency\textsuperscript{14}

Figure: The 1\textsuperscript{st} controller can oper 5 modifications before provoking a synchronisation. The second has 3 operations. The system should update from this flow...
Examples: Adaptive consistency\textsuperscript{14}

Examples: Adaptive consistency\textsuperscript{14}

Figure: First, add a new rule. $C_1$ has consumed 1 operation.

Examples: Adaptive consistency\textsuperscript{14}

\textbf{Figure}: And another (2 operations used).

\textsuperscript{14} Sakic et al. “Towards Adaptive State Consistency in Distributed SDN Control
Examples: Adaptive consistency\textsuperscript{14}

Examples: Adaptive consistency\textsuperscript{14}

\textbf{Figure:} And A fouth rule.

Examples: Adaptive consistency

Figure: Remove a rule. $C_1$ has consumed 5 operations. Hence triggers a synchronisation.
Examples: Adaptive consistency\textsuperscript{14}

\textbf{Figure:} Synch step done!

State of the art and challenges on consistency management at switch and controller layers

On the control plane

Consistency in Controllers Version Update

Controllers Version Update

What does it mean?

- Some unexpected events can modify the dataplane
- Hence, it creates inconsistencies between the version of the network that the controller has and the true network.
- Can create forwarding loops, blackholes...
- Can happen during an update.

---

Example

Figure: During an update of the rules...
Example

Figure: ...a switch goes down...
Example

Figure: …BOOM! A black hole!
What kind of system to detect and fix it?
State of the art and challenges on consistency management at switch and controller layers

On the control plane

Consistency in Controllers Version Update

Solutions?

**Detection:** Header Space Analysis\(^{16}\), VeriFlow\(^{17}\)

**Solving inconsistencies:** OFRewind\(^{18}\), HotSwap\(^{19}\), multi-commits transactional semantics\(^{20}\).

**Multi-commits transactional semantics:** A consistent message processing. Being able to rollback. Each transaction is splitted in subtransactions, and checks are performed in order to avoid inconsistencies between sub-transactions. At the end of a transaction, it is committed if all sub-transactions are read, or if there is no read-write conflict.

---


\(^{19}\) Vanhever et al. “HotSwap: Correct and Efficient Controller Upgrades for

---

21/27
State of the art and challenges on consistency management at switch and controller layers

On the control plane

Consistency in Controllers Version Update

**Figure:** Interaction between transactions. $T_1$’s write conflicts with already committed $T_2$’s read. Hence, $T_1$ must be aborted otherwise it would create an inconsistency.
Conclusion

- Many problems on both control and data plane.
- Some are already addressed in ONOS.
- Solutions already exist for some other problems.
References


References


