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

Rémi Oudin, LIP6

April 10, 2018

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.

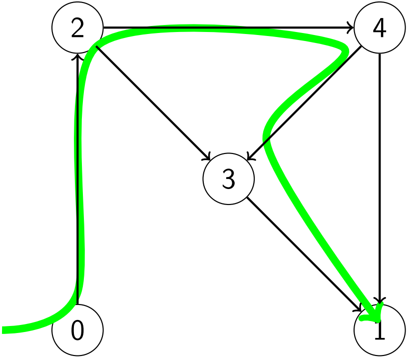
Rules update consistency: Data plane

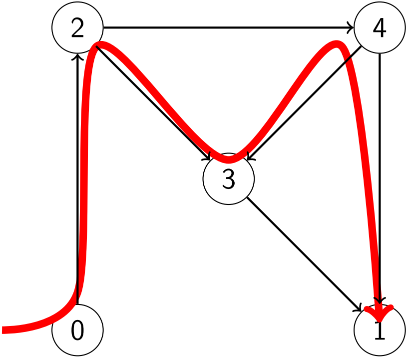
Consistent Network Update

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

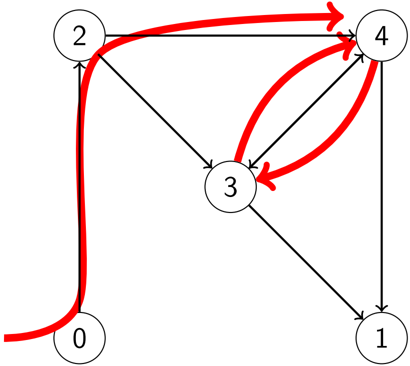
General statement

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.


2
4
3
0
1


2
4
3
0
1

What if we add first a rule from (3) to (4) ?


2
4
3
0
1

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

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.

[2] Foerster et al. "Loop-Free Route Updates for Software-Defined Networks" in, 2017.

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).

---

[3] Reitblatt et al. "Abstractions for Network Update" in *SIGCOMM '12*, 2012.

[4] Vissicchio et al. "Safe Update of Hybrid SDN Networks" in *IEEE/ACM Transactions on Networking*, 2017.

[5] McGeer. "A Correct, Zero-overhead Protocol for Network Updates" in *HotSDN '13*, 2013.

[6] Ludwig et al. "Good Network Updates for Bad Packets: Waypoint Enforcement Beyond Destination-Based Routing Policies" in *HotNets-XIII*, 2014.

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.

---

[7] Liu et al. "zUpdate: Updating Data Center Networks with Zero Loss" in *SIGCOMM Comput. Commun. Rev.* 2013.

[8] Zheng et al. "Minimizing Transient Congestion during Network Update in Data Centers" in, 2015.

[9] Reitblatt et al. "Abstractions for Network Update" in *SIGCOMM '12*, 2012.

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.

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.

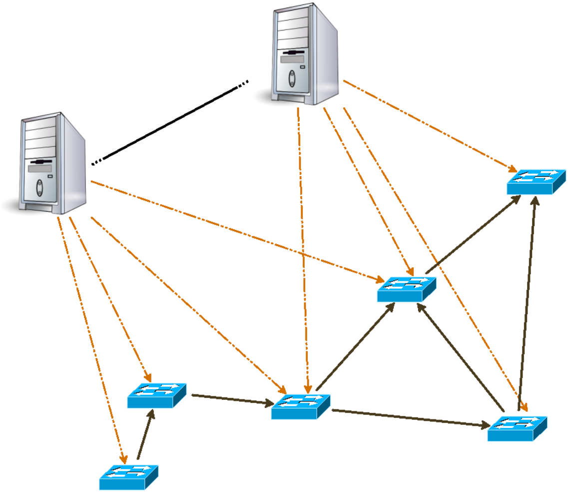


Figure: A multiple controller SDN network...

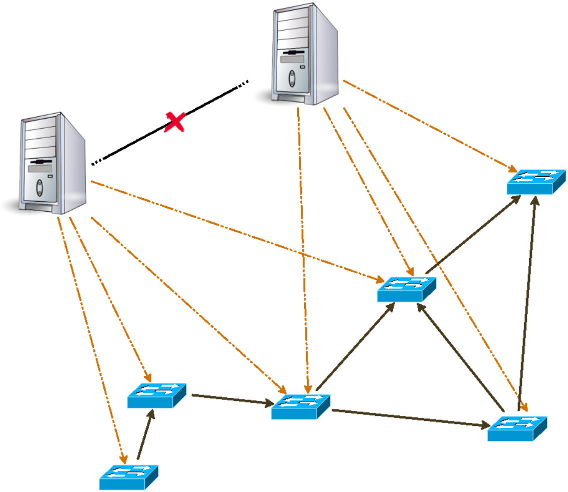


Figure: . . . with a control plane link down

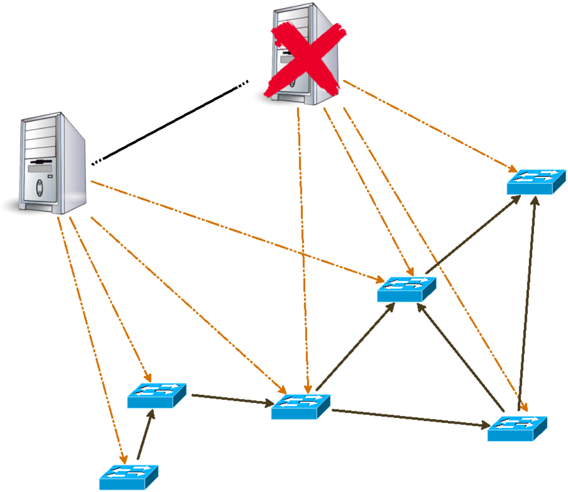


Figure: . . . with a controller down

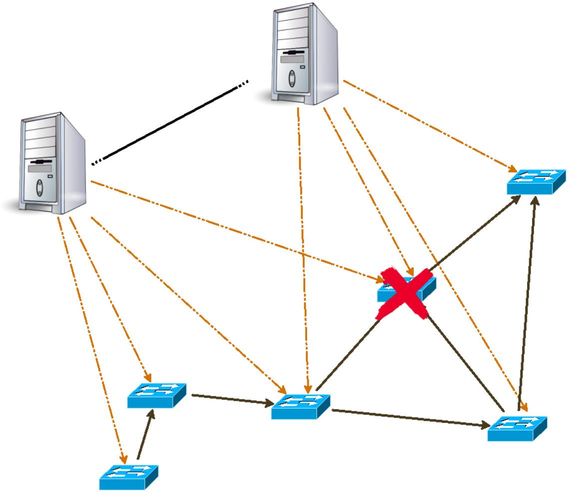


Figure: . . . with a switch down

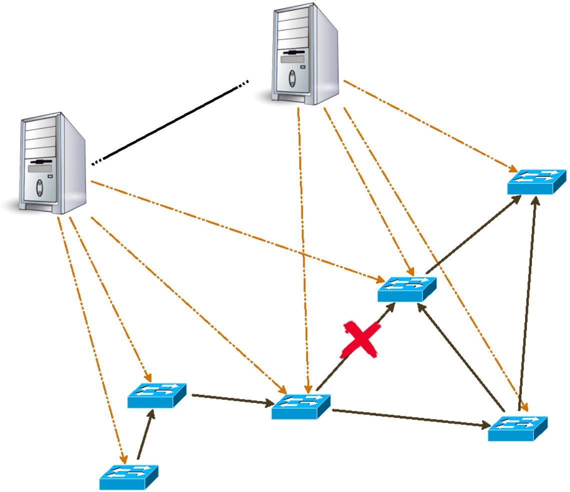


Figure: . . . with a data plane link down

Consistencies for state consistency

Strong consistency[11]: Slow, CPU intensive. Sync between each operation.

↪ Exists in ONOS: Raft algorithm

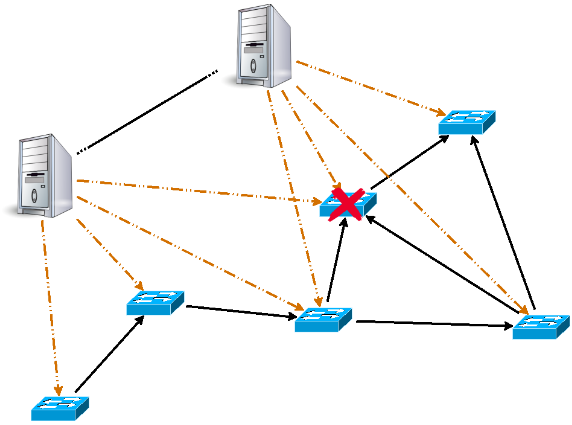
Eventual consistency: Fast, but reliable mostly if few writes are done. Guarantees that if no new updates are made, all accesses will eventually return the last updated value.

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

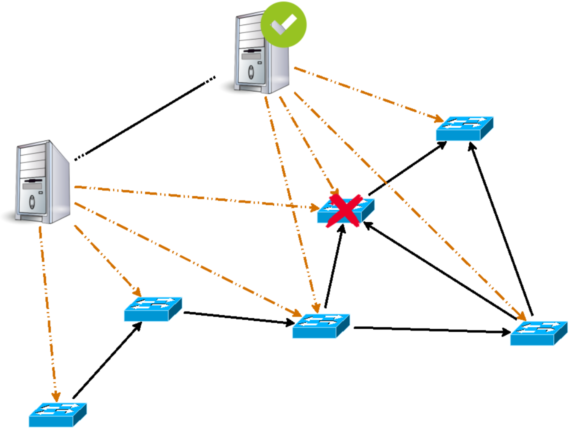
Adaptive consistency[12]: Consistency level is adapted according to the load (read and writes). It can go from strong down to eventual consistency.
«State synchronisation occurs according to performance and consistency constraints set by the application at runtime.»

[11] Botelho et al. "On the Feasibility of a Consistent and Fault-Tolerant Data Store for SDNs" in, 2013.

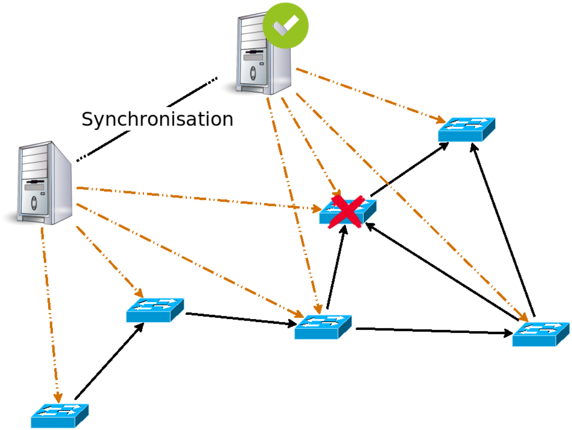
Examples: Strong consistency



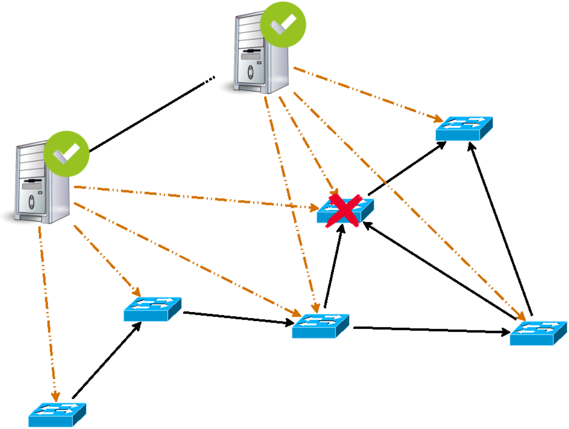
Examples: Strong consistency



Examples: Strong consistency



Examples: Strong consistency



Examples: Eventual Consistency

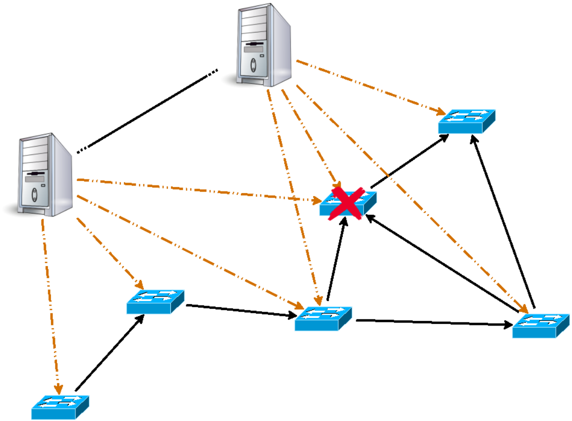


Figure: At some point, a switch goes down, or a link is broken.

Examples: Eventual Consistency

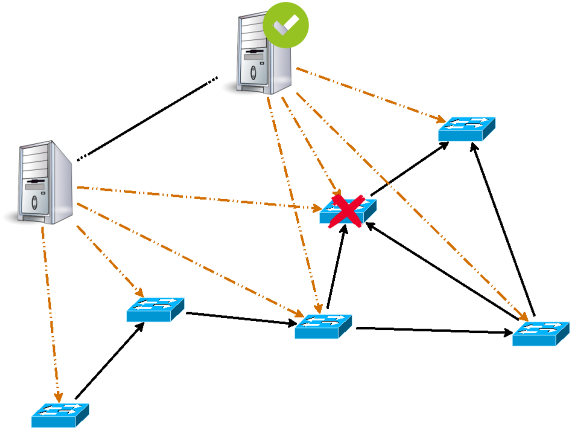


Figure: The controller detects the problem.

Examples: Eventual Consistency

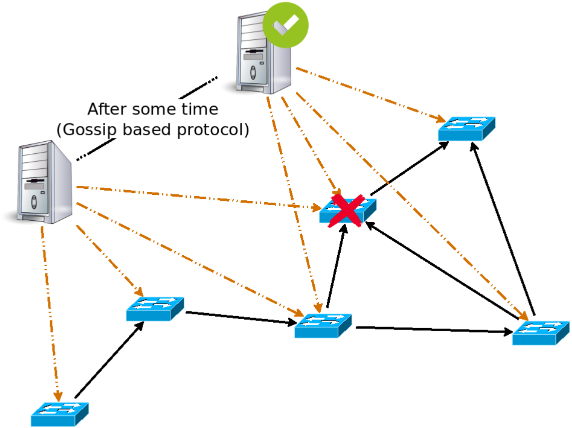


Figure: The other controller is *eventually* notified after a repair protocol. (Ex : Gossip based protocol, fix-on-read, fix-on-write)

Examples: Eventual Consistency

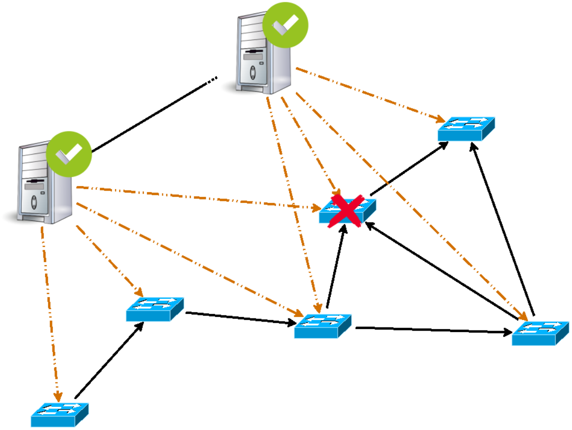


Figure: The controller is notified from the change, state consistency is restored.

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.

[13] Sakic et al. “Towards Adaptive State Consistency in Distributed SDN Control Plane” in, 2017.

Examples: Adaptive consistency[14]

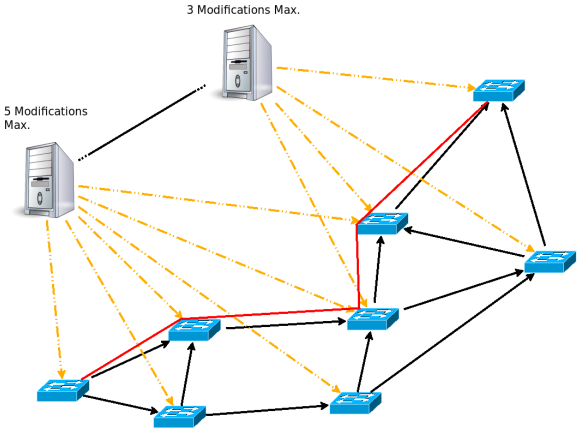


Figure: The 1st controller can oper 5 modifications before provoking a synchronisation. The second has 3 operations. The system should update from this flow. . .

Examples: Adaptive consistency[14]

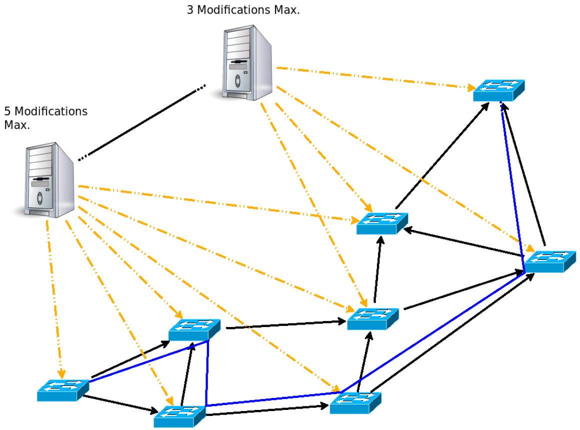


Figure: . . . to this one.

[14] Sakic et al. "Towards Adaptive State Consistency in Distributed SDN Control

Examples: Adaptive consistency[14]

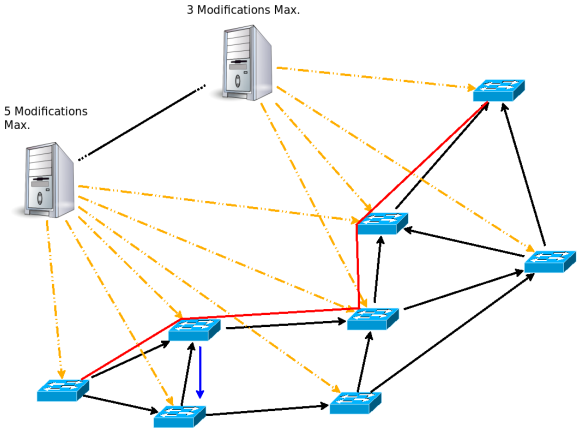


Figure: First, add a new rule. C_1 has consumed 1 operation.

[14] Sakic et al. "Towards Adaptive State Consistency in Distributed SDN Control

Examples: Adaptive consistency[14]

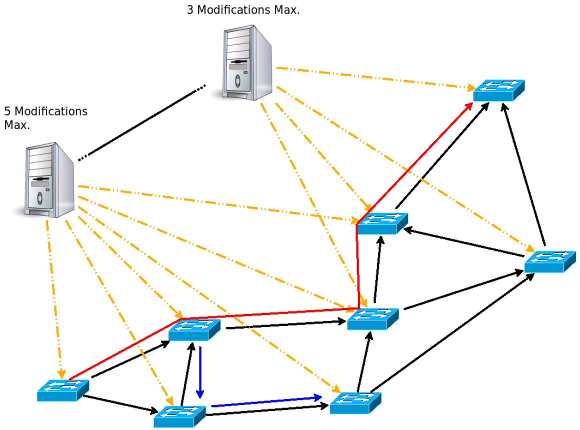


Figure: And another (2 operations used).

[14] Sakic et al. "Towards Adaptive State Consistency in Distributed SDN Control

Examples: Adaptive consistency[14]

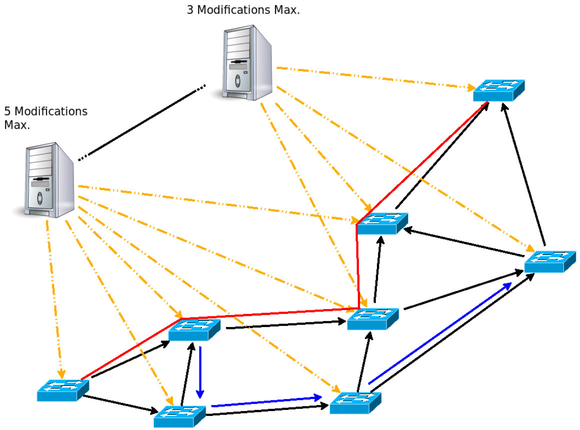


Figure: And another.

[14] Sakic et al. “Towards Adaptive State Consistency in Distributed SDN Control

Examples: Adaptive consistency[14]

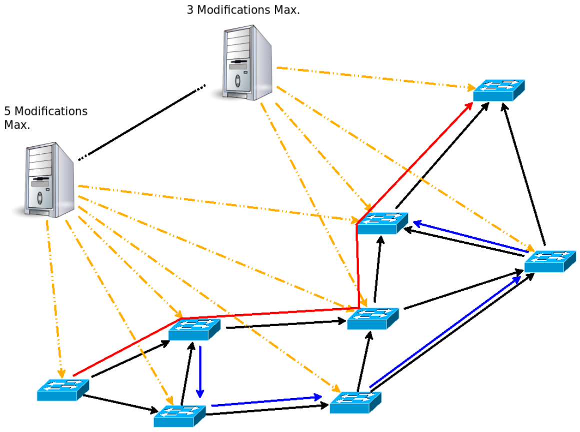


Figure: And A fouth rule.

[14] Sakic et al. "Towards Adaptive State Consistency in Distributed SDN Control

Examples: Adaptive consistency[14]

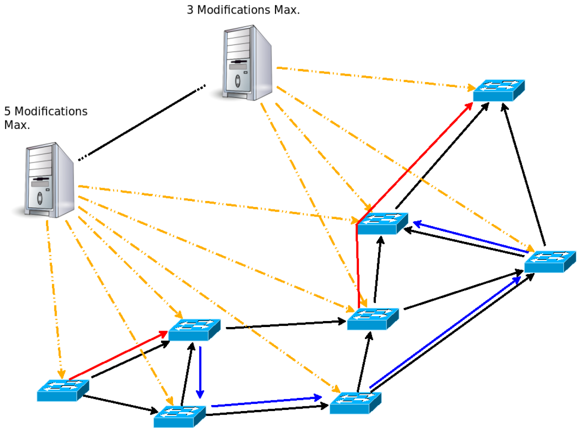


Figure: Remove a rule. C_1 has consumed 5 operations. Hence triggers a synchronisation.

Examples: Adaptive consistency[14]

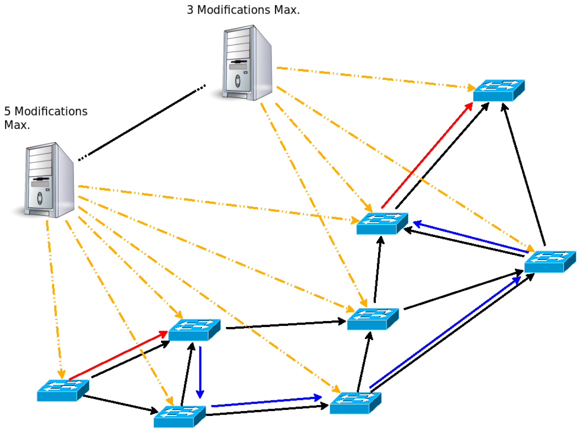


Figure: Synch step done !

[14] Sakic et al. "Towards Adaptive State Consistency in Distributed SDN Control

Controllers Version Update[15]

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.

[15] Kazemanian, Varghese, and McKeown. "Header Space Analysis: Static Checking for Networks" in *Network System Design and Implementation (NSDI)*, 2012.

Example

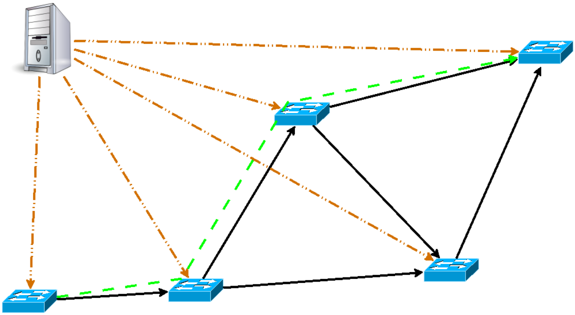


Figure: During an update of the rules. . .

Example

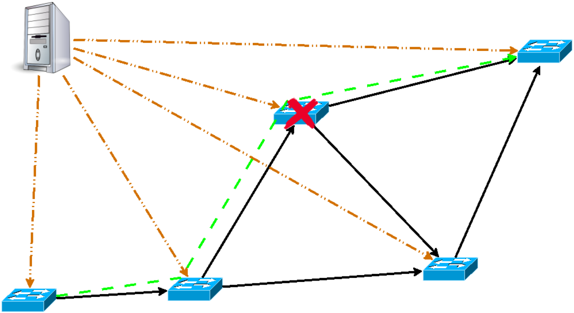


Figure: . . . a switch goes down. . .

Example

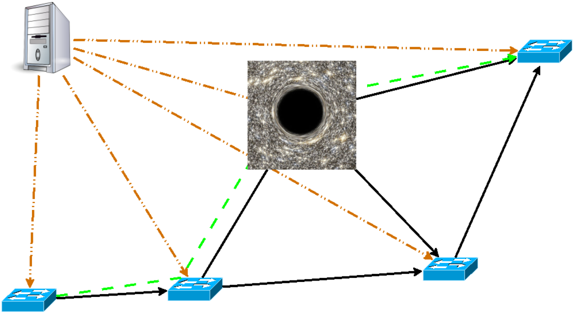


Figure: . . . BOOM! A black hole!

Example

What kind of system to detect and fix it ?

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.

---

16 Kazemanian, Varghese, and McKeown. "Header Space Analysis: Static Checking for Networks" in *Network System Design and Implementation (NSDI)*, 2012.

17 Khurshid et al. "VeriFlow: Verifying Network-Wide Invariants in Real Time" in, 2013.

18 Wundsam et al. "OFRewind: enabling record and replay troubleshooting for networks" in, 2011.

19 Vanbever et al. "HotSwap: Correct and Efficient Controller Upgrades for

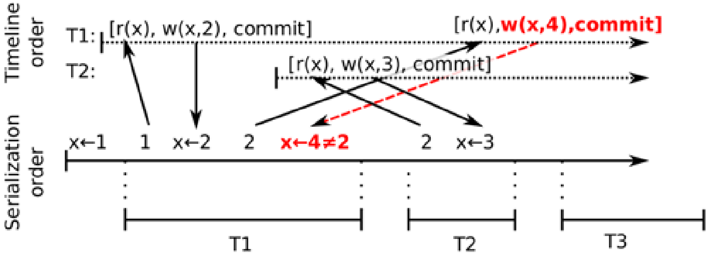


Figure: Interaction between transactions. T_1's write conflitcs with already committed T_2's read. Hence, T_1 must be aborted otherwise it would create an inconsistency

Conclusion

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

- M. Aslan and A. Matrawy. "Adaptive consistency for distributed SDN controllers". In: *2016 17th International Telecommunications Network Strategy and Planning Symposium (Networks)*. 2016, pp. 150–157. DOI: 10.1109/NETWKS.2016.7751168.
- Othmane Blial, Mouad Ben Mamoun, and Benaini Redouane. "An Overview on SDN Architectures with Multiple Controllers". In: 2016 (Jan. 2016), pp. 1–8.
- F. A. Botelho et al. "On the Feasibility of a Consistent and Fault-Tolerant Data Store for SDNs". In: *2013 Second European Workshop on Software Defined Networks*. 2013, pp. 38–43. DOI: 10.1109/EWSDN.2013.13.
- Klaus-Tycho Foerster et al. "Loop-Free Route Updates for Software-Defined Networks". In: PP (Dec. 2017), pp. 1–14.
- Paymen Kazemanian, George Varghese, and Nick McKeown. "Header Space Analysis: Static Checking for Networks". In: USENIX, 2012. URL: https://www.microsoft.com/en-us/research/publication/header-space-analysis-static-checking-for-networks/.

Ahmed Khurshid et al. "VeriFlow: Verifying Network-Wide Invariants in Real Time". In: *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*. Lombard, IL: USENIX, 2013, pp. 15–27. ISBN: 978-1-931971-00-3. URL: https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/khurshid.

Hongqiang Harry Liu et al. "zUpdate: Updating Data Center Networks with Zero Loss". In: *SIGCOMM Comput. Commun. Rev.* 43.4 (Aug. 2013), pp. 411–422. ISSN: 0146-4833. DOI: 10.1145/2534169.2486005. URL: http://doi.acm.org/10.1145/2534169.2486005.

Arne Ludwig et al. "Good Network Updates for Bad Packets: Waypoint Enforcement Beyond Destination-Based Routing Policies". In: *Proceedings of the 13th ACM Workshop on Hot Topics in Networks. HotNets-XIII*. Los Angeles, CA, USA: ACM, 2014, 15:1–15:7. ISBN: 978-1-4503-3256-9. DOI: 10.1145/2670518.2673873. URL: http://doi.acm.org/10.1145/2670518.2673873.

Rick McGeer. "A Correct, Zero-overhead Protocol for Network Updates". In: *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. HotSDN '13*. Hong Kong, China: ACM, 2013, pp. 161–162. ISBN: 978-1-4503-2178-5. DOI: 10.1145/2491185.2491217. URL: http://doi.acm.org/10.1145/2491185.2491217.

Peter Perešíni et al. "OF.CPP: Consistent Packet Processing for Openflow". In: *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*. *HotSDN '13*. Hong Kong, China: ACM, 2013, pp. 97–102. ISBN: 978-1-4503-2178-5. DOI: 10.1145/2491185.2491205. URL: http://doi.acm.org/10.1145/2491185.2491205.

Mark Reitblatt et al. "Abstractions for Network Update". In: *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*. *SIGCOMM '12*. Helsinki, Finland: ACM, 2012, pp. 323–334. ISBN: 978-1-4503-1419-0. DOI: 10.1145/2342356.2342427. URL: http://doi.acm.org/10.1145/2342356.2342427.

Ermin Sakic et al. "Towards Adaptive State Consistency in Distributed SDN Control Plane". In: *IEEE International Conference on Communications (ICC)*. Paris, France, 2017.

Laurent Vanbever et al. "HotSwap: Correct and Efficient Controller Upgrades for Software-defined Networks". In: *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*. *HotSDN '13*. Hong Kong, China: ACM, 2013, pp. 133–138. ISBN: 978-1-4503-2178-5. DOI: 10.1145/2491185.2491194. URL: http://doi.acm.org/10.1145/2491185.2491194.

S. Vissicchio et al. “Safe Update of Hybrid SDN Networks”. In: *IEEE/ACM Transactions on Networking* 25.3 (2017), pp. 1649–1662. ISSN: 1063-6692. DOI: 10.1109/TNET.2016.2642586.

Andreas Wundsam et al. “OFRewind: enabling record and replay troubleshooting for networks”. In: *In Proceedings of USENIXATC'11*. 2011.

Yuan Zhang et al. “A survey on software defined networking with multiple controllers”. In: *Journal of Network and Computer Applications* 103 (2018), pp. 101 –118. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2017.11.015. URL: http://www.sciencedirect.com/science/article/pii/S1084804517303934.

J. Zheng et al. “Minimizing Transient Congestion during Network Update in Data Centers”. In: *2015 IEEE 23rd International Conference on Network Protocols (ICNP)*. 2015, pp. 1–10. DOI: 10.1109/ICNP.2015.33.