Sridut: Securing Multi-Controller SDN Integrity with State-aware Multi-consistency Storage and Provable Convergence

Adir Miller, Avinash Singh and Hein Venter
Department of Computer Science, University of Pretoria, South Africa

adir.miller@tuks.co.za
asingh@cs.up.ac.za
hventer@cs.up.ac.za

Abstract: Software Defined Networking (SDN) is a network paradigm that decouples the control and data planes, enabling centralized network control. The single centralized controller of the original paradigm, however, created a single point of failure, which was overcome with the introduction of multi-controller networks. These networks, however, introduce new security challenges as the availability and integrity of the networks now depend on state consistency mechanisms and their inherent trade-offs. For example, multi-controller networks that solely support strong consistency enforce correctness at the cost of availability due to their susceptibility to Denial of Service (DoS) attacks and maliciously induced partitions. Alternatively, some designs employ hybrid consistency or multi-consistency, sacrificing correctness guarantees to strike a balance between correctness and availability. However, these designs lack mechanisms for provably verifiable convergence, resulting in critical integrity violations such as corrupted and conflicting security and Quality of Service (QoS) policies. This critical gap is exploitable by adversaries, enabling them to trigger policy conflicts and bypass security perimeters. These security implications are further amplified by the lack of transparency between the application layer and consistency state events. This transparency hampers the ability of the controller to react and recover from consistency-based attacks. To address these challenges, this paper introduces Sridut, a secure multi-consistency storage model providing strong consistency, strong eventual consistency, and consistency state awareness. The model provides multiple storage backends with varying degrees of consistency guarantees, allowing for more granular control of the inherent security trade-offs. Additionally, the consistency state, health, and critical events are made transparent, further improving an application's ability to react to consistency-based attacks. The model leverages Conflict-free Replicated Data Types (CRDTs) and anti-entropy mechanisms to achieve strong eventual consistency. This mechanism allows divergent states to be safely merged, preserving integrity of critical data such as network policies. Consequently, this approach mitigates exploitable security risks associated with divergent states. Ultimately, the state-aware architecture and provable convergence of Sridut provide a robust defence against both malicious threats and inherent network partitions, helping preserve integrity and availability in SDN multi-controller networks.

Keywords: Software-Defined Networking (SDN), Conflict-free Replicated Data Types (CRDTs), Network security, Multi-controller architecture, Consistency state-awareness

1. Introduction

As Software-Defined Networking (SDN) becomes the backbone of critical infrastructure, the security and resilience of its control plane have become paramount (Yusuf et al, 2023). While multi controller networks mitigate the single point of failure found in early implementations of SDN networks, the transition introduces vulnerabilities within the consistency protocols used to manage shared data (Zhang et al, 2018). Attackers can exploit trade-offs inherent in these consistency protocols to trigger policy conflicts or bypass security perimeters (Xu, 2017). Additionally, many controllers deny the application layer the transparency needed to assess the state and health of the consistency stores, effectively removing the applications ability to assist and react to consistency-based attacks (Sakic and Kellerer, 2018). Furthermore, while some modern SDN controllers offer various levels of consistency models (Smyth et al, 2019), weaker consistency models are often less likely to be used due to their lack of verifiable convergence.

To address these challenges, this paper introduces Sridut, a secure multi-consistency storage model targeting three consistency aspects: strong consistency, strong eventual consistency, and consistency state awareness. Sridut functions as a unified storage broker that abstracts individual consistency backends, as well as providing a transparent API for critical state events. These events are critical for providing security applications and allowing for the immediate detection and mitigation of consistency-related vulnerabilities. By leveraging Conflict Free Replicated Data-Types (CRDTs) (Shapiro et al, 2011) and anti-entropy mechanisms, Sridut allows for weaker consistency models to be used in areas it would otherwise be unsuitable. This allows for the high availability during network partitions without sacrificing control plane integrity, ensuring that critical security and QoS policies are deterministically merged rather than overwritten or lost. The paper is structured as follows, Section 2 provides a brief background into the possible security issues of SDN multi-controller networks, an introduction to CRDTs and related works. Section 3 presents the Sridut model. Section 4

introduces the Proof of Concept, as well as a use case scenario comparing the PoC to a well-known SDN controller. Lastly, Section 5 concludes the paper and presents future research opportunities.

2. Background

Consistency is a property that governs how data appears and behaves across the nodes of a distributed system. This property is governed by inherent trade-offs as defined by the CAP theorem (Brewer, 2000), which states that during a partition, it is impossible to guarantee both consistency and availability simultaneously (Brewer, 2000). In an SDN multi-controller network, strong consistency is typically utilized to enforce a unified global policy (Zhang et al, 2018). However, solely utilizing strong consistency creates a significant risk to availability (Shapiro et al, 2011); by targeting the consensus mechanism, an attacker can prevent the controllers from making progress and freeze the network (Smyth et al, 2019). Conversely, solely utilizing eventual consistency allows an attacker to target the windows of divergence and bypass crucial security policies (Xu, 2017), creating a threat to network integrity. Additionally, the convergence of these divergent states often leads to conflicts that need to be resolved. If resolved non-deterministically, it may lead to inconsistent or unintended policy states, which can be leveraged by an attacker to corrupt critical network policies (Xu, 2017). While the CAP theorem is often treated as a binary trade-off, it is more accurately described as a sliding scale or spectrum between availability and consistency (Brewer, 2012). When applied to SDN, it is beneficial for controllers to provide different levels of consistency for different data, or adaptive consistency models that tailor the consistency level factors such as workload, latency, and network conditions (Bannour, Souihi, and Mellouk, 2020). Existing SDN controllers implement multi-consistency, however, they suffer from significant flaws such as the lack of transparency, the lack of verifiable convergence, and the pre-determination of consistency levels (Sakic and Kellerer, 2018; Xu, 2017; Hoang et al, 2022). The lack of verifiable convergence on weak consistency models creates room for an attacker to bypass security policies by exploiting the divergent states. The lack of transparency to the health of the network as well as the state of the consistency stores effectively masks security applications from reacting to consistency-based attacks. Furthermore, despite supporting multi-consistency, many controllers still prioritize consistency during a partition (Yusuf et al, 2023), making the controller unsafe to utilize for networks which require high availability. SDN is a generalized, adaptable network paradigm, meaning it needs to be applicable to different industries with different threat models. Therefore, the choice of consistency cannot remain a static controller-side configuration and instead must be moved to the application layer.

Current SDN controllers do not provide sufficient consistency state transparency, configurability of consistency, and lack weak consistency stores with verifiable convergence. This creates a critical gap which prevents security applications from assessing system integrity or responding to consistency-related attacks. Additionally, the lack of verifiable convergence results in possible policy violations and discourages the use of weaker consistency models for critical states, forcing reliance on strong consistency and its associated availability limitations.

2.1 Conflict Free Replicated Data-types

Conflict-free Replicated Data Types (CRDTs) (Shapiro et al, 2011) are a group of distributed data structures that can guarantee deterministic convergence without the need for coordination or conflict recovery. The ability to work without coordination allows CRDTs to maintain correctness in unstable network environments as well as network partitions. Additionally, while clock drift and message ordering may affect when updates are applied, the final converged state will remain consistent with no data loss across all nodes. These features help prevent attackers from utilizing windows of state divergence and partitions to corrupt or bypass network security policies. CRDTs are generally split into two general groups: operation-based (Op-CRDTs), which propagate operations, and state-based (State-CRDTs), which propagate the entire state with merge semantics. δ-CRDTs (Almeida et al, 2015) are an optimized subset of the state-CRDTs that only propagate the incremental changes, reducing communication overhead while preserving convergence.

2.2 Related Work

Adaptive consistency models have been proposed to balance the trade-off between performance and accuracy in SDN. Sakic and Kellerer (2018) presents an adaptive consistency which utilizes a custom metric to allocate the required consistency level. This metric assesses the approximation ratio between optimal serialized decisions, and the actual observed results of the network. Additionally, the work utilises CRDTs to ensure that the adaptive consistency store maintains verifiable convergence. In addition to the adaptive store, the work incorporates a dedicated strong and eventual store. However, the work places a higher focus on performance

optimization instead of security. Additionally, the work considers a benign fail-recovery environment with a fair control channel and does not explore the effects of malicious network partitions and unreliable links.

While SDN is a generalized paradigm, specialized frameworks tailored to specific industries have been explored. Bannour, Souihi, and Mellouk (2020) present an adaptive consistency approach for SDN that is tailored to Content-Centric Delivery Network (C-CDN) in large-scale Internet of Things (IoT) environments. The work replaces the eventually consistent store with an adaptive quorum-replicated consistency model. This model can adjust quorum sizes dynamically to strike a balance between correctness and availability. However, the proposed solution focuses on efficiency and does not consider the security trade-offs inherent in adaptive coordination.

While some consistency mechanisms may be mathematically sound, the implementation might introduce unintended side effects that leave the system open to exploitation. Therefore, Kim et al (2024) present Ambusher, a generic testing tool utilizing protocol state fuzzing to identify critical vulnerabilities in the east-west interface of SDN multi controller networks. The work infers a state machine by analyzing transitions across the leadership, membership, mastership, and distributed storage engines. Using this state machine, Ambusher can detect undiscovered vulnerabilities in well-known controllers. While Ambusher serves as a diagnostic testing tool, it displays how the transparency of the internal cluster state can be used to improve control plane security.

3. Sridut

To allow for stronger, more robust SDN multi-controller networks, the work introduces the Sridut model, named after the Hebrew word for survivability, שְׂרִידוּת. As shown in Figure 1, Sridut contains a unified storage broker that provides strong consistency, strong eventual consistency, and consistency state awareness. Sridut contains four different registries, each with unique properties and a single point of entry known as the State Access Layer (SAL). The first two registries are the strong and CRDT registries, used when data needs to be shared across multiple controllers in a network. To reduce the load on both the strong and CRDT stores, Sridut also contains two additional stores that can be used by applications for data that is local to a specific controller. The in-memory registry provides a fast, non-persistent store local to a controller, whereas the local store provides a persistent store local to the controller. The remaining stores will be covered in more detail in the following section. Sridut contains a dedicated membership module used to keep track of the active controllers in the network. The two components of an SDN controller that interface with Sridut are the Northbound Interface and the Core services.

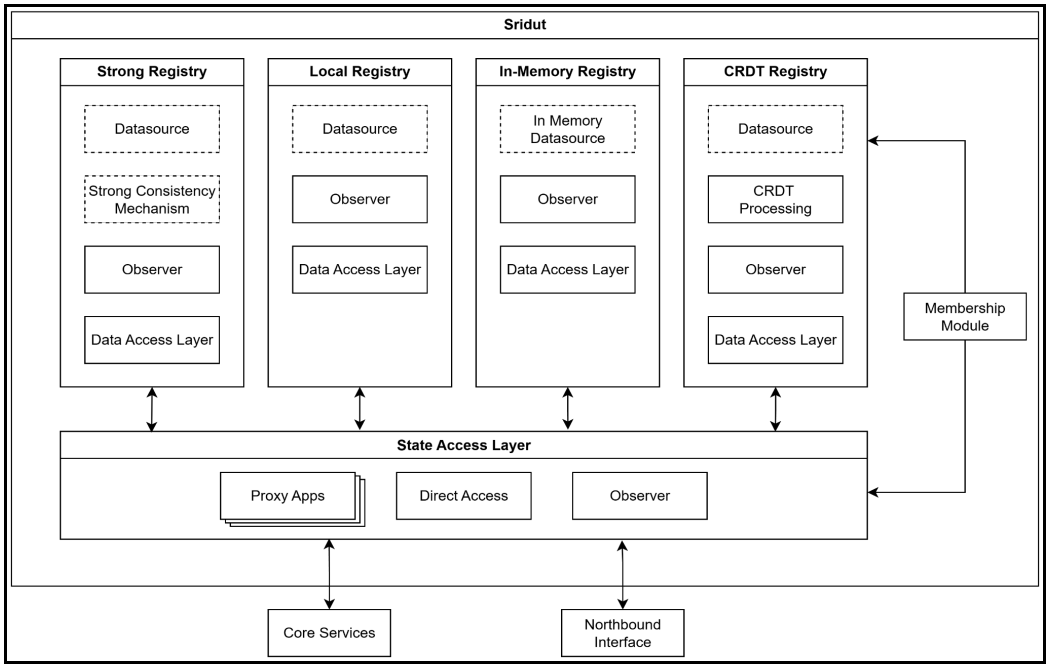

Figure 1: High level overview of Sridut

3.1 Common Components

All registries have three common components, an observer, a Data Access Layer (DAL) and a datasource. The datasource represents the physical storage backend where the data resides. For all registries barring the in-memory registry, this datasource needs to be persistent to ensure data does not get lost after a controller crash. Each registry contains an observer which allows critical network changes such as IP bans to be pushed out rapidly. This ability to react rapidly to network events is critical for ensuring network security in a highly dynamic environment. The DAL ensures that the applications work off an abstracted view of the stores, instead of being exposed to the underlying storage methodology. This creates a separation of concerns and helps prevent technical debt, as newer, safer stores can be utilized without needing to rewrite the application. Furthermore, this can help shield applications from the complexities and possible security risks of direct datasource interaction since these complexities are abstracted behind a consistent interface.

3.2 Membership Module

The ability for a controller to keep track of the other controllers and their health in the network is paramount to ensuring the controllers can operate as a single logical unit. The membership Module of Sridut is based on the Scalable Weakly Consistent Infection-style Process Group Membership (SWIM) protocol (Das, Gupta and Motivala, 2002). SWIM provides decentralized gossip-based membership with built-in failure detection. This grants applications transparency into the health and status of all controllers in a network, allowing for applications to react in real time to possible attacks.

3.3 Strong Registry

In normal operations using strong consistency in SDN multi-controller networks can help prevent conflicts, loops, and security policy violations (Xu, 2017). Figure 2 presents the internal layout of the strong consistency store used by Sridut. The Observer receives updates on the state health of the consensus mechanism, such as quorum losses, leadership changes and liveness. This allows other components the ability to react rapidly to failures or attacks on the strong consistency mechanism.

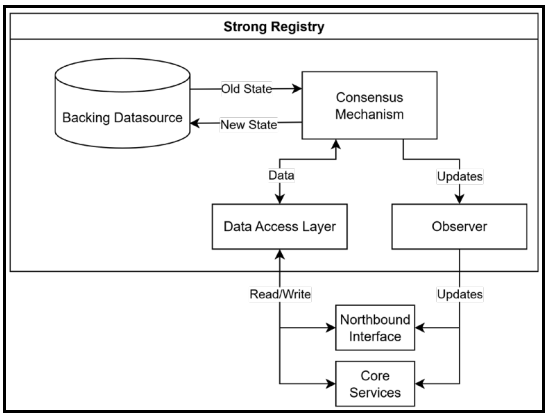

Figure 2: The registry that provides strong consistency

3.4 CRDT Registry

The CRDT Registry (Figure 3) provides strong eventual consistency by leveraging the verifiable convergence of CRDTs and the state reconciliation capabilities of an anti-entropy protocol.

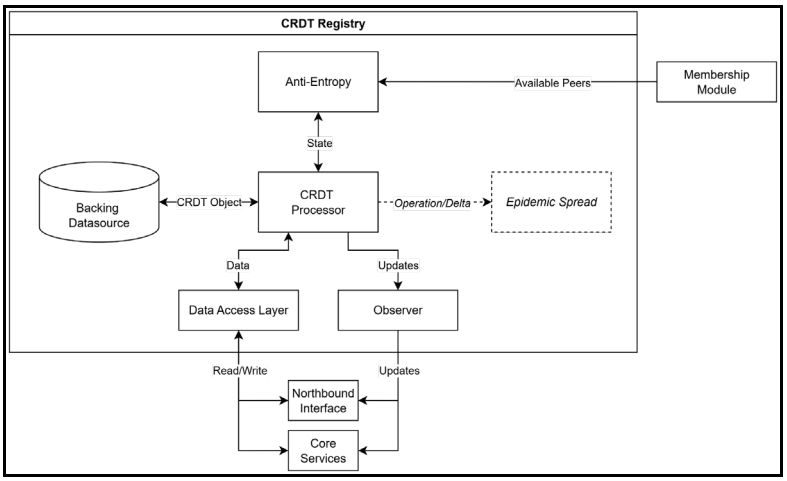

Figure 3: The registry that provides strong eventual consistency through CRDTs

The DAL abstracts out the low-level metadata that CRDTs use to function correctly, reducing complexity for applications when accessing and manipulating CRDT objects. The CRDT Registry should not be limited to one type of CRDT and instead should offer multiple different CRDT types such as PN-Counter, G-Sets and Or-Sets. The CRDT processor is the core of the CRDT Registry, handling all merge operations, irrespective of the type of CRDT used. The Backing Datasource stores the CRDT objects with all required metadata. The anti-entropy mechanism ensures that all nodes eventually converge in the case that updates are missed. To ensure all applications are kept up to date, and to ensure critical updates are applied as fast as possible, the CRDT Processor pushes updates to the observer, which pushes updates downstream. The Epidemic Spread module is an optional module that pushes out updates in a gossip style fanout, ensuring that critical network updates get spread through the network rapidly. This is crucial for security applications time-sensitive operations, such as IP bans, where rapid propagation helps minimize potential damage in the event of an attack.

3.5 State Access Layer

The State Access Layer (SAL) acts as a unified interface allowing applications and core network services to interact with the various registries. The observer module allows applications to receive events from multiple registries through a single unified stream, allowing for the detection of possible security issues across the entire control plane. The Direct Access component grants applications the ability to directly communicate with the various DALs of the registries, allowing applications the ability to easily choose the required level of consistency.

SDN has been used extensively to introduce more advanced security to networks through its programmable and customizable nature (Yusuf et al, 2023; Zhang et al, 2018). Proxy Apps utilize a similar principle and add programmability between the applications and the storage layer of SDN networks. The design of Sridut allows for these proxies to be created that act as middleware or as smart intermediaries. For example, a proxy application could use the available stores to simulate adaptive consistency, choosing the right consistency store based on the data and available metrics.

4. Evaluation

To evaluate Sridut, a proof of concept was developed and tested against ONOS, a popular industry grade SDN controller. Evaluating Sridut directly is challenging, as the model defines the underlying primitives for security rather than the end-user logic. Therefore, an application called HighAvail, was developed, which uses the primitives to achieve a greater level of network security.

4.1 Proof of Concept

The Proof of Concept (PoC) was built on top of os-ken (Openstack, 2025) SDN controller. A global registry singleton is made available to both applications and to the core network services. The global observer of the SAL is implemented with a Pymitter (Rieger, 2025) event emitter and was held in a singleton. The Strong Registry utilizes ETCD (CNCF, 2025), a strongly consistent key value store based on the Raft (Ongaro and Ousterhout, 2014) consensus algorithm. The Local Registry utilizes the built in Python shelve module, which allows persistent python objects stored in a binary file. The In-Memory Registry utilizes standard python in memory data structures such as maps and arrays. The CRDT Registry is built on Pycrdt (Brochart, 2025), a

Python library providing δ-CRDTs. The optional ESM module was implemented using push gossip with fanout, TTL, and duplicate suppression. Lastly the Membership Module utilizes Serf (Hashicorp ,2025), a membership protocol based on SWIM.

To evaluate Sridut, it is necessary to explore how an application can use the various primitives provided to enhance the security of the control plane. For this reason, the proxy application HighAvail was developed. HighAvail provides strong consistency in normal operation while falling back to strong eventual consistency during unplanned events like a partition. During normal operation, anti-entropy is disabled, and each write to both the strong and eventual stores. This ensures that there is always adequate CRDT metadata in case of a partition. When a partition is detected by the SAL observer, the minority side(s) enable anti-entropy and forward all reads and writes to the CRDT store. A key difficulty in multi-consistency systems is ensuring no changes are lost during recovery. HighAvail, however, uses the verifiable convergence of CRDTs to ensure no updates are lost. Once the partition is healed, the minority side does not immediately swap over to the strong store. Instead, the system first exploits the mathematical guarantees of CRDTs to ensure all nodes converge to the same CRDT state. After a partition is healed, the minority nodes send out a signal indicating they are in a fallback mode, causing all anti-entropy stores to be enabled. During recovery, a deterministic leader reconciles divergent states, and minority nodes independently verify that their local CRDT state is identical to the strong store and return to utilizing the strong store. Once all nodes transfer back to utilizing the strong store, anti-entropy is disabled.

4.2 Testbed

To accurately evaluate the PoC against currently available controllers, a testbed is required to run repeatable and reproducible tests. The data plane is simulated by Mininet (Lantz, Heller and McKeown, 2010), a popular SDN emulator widely utilized in academia. While it is possible to use Mininet to simulate the entire network, there are limited options for emulating the control plane itself. Therefore, control plane emulation is handled through Containerlab (Srl-labs, 2025), a container-native orchestration tool specially designed for lab environments. The testbed consisted of a virtual machine with 6 vCPUs (AMD Ryzen 5800X @ 3.8 GHz), 8GB DDR4 RAM, running Ubuntu 20.04.6 LTS, utilizing ONOS v2.7.0, Containerlab v0.69.3, and Mininet v2.3.0.dev6.

4.3 Use Case

This section covers a use case scenario showing how Sridut allows for stronger network security through the verifiable convergence and consistency state awareness. As a baseline for the use case scenario, the work makes use of the industry standard ONOS (Berde et al, 2014) SDN controller. Figure 4 depicts an SDN network topology with a control plane using out-of-bound communication. In this topology, there is a malicious or infected host, h2s2. A split-brain partition can be induced by removing the link from CP Switch 1 and CP Switch 2. Although the controllers can technically communicate through S1 and S2, the testbed is modelled using out-bound communication between controllers which would not allow this form of communication. Using this topology two identical sets of experiments were conducted, with the same sequence of steps. In both cases a basic Access Control List (ACL) application was used for both the PoC and ONOS. The results of the experiment can be seen in Table 1. The PoC uses a HighAvail store to store ACL rules.

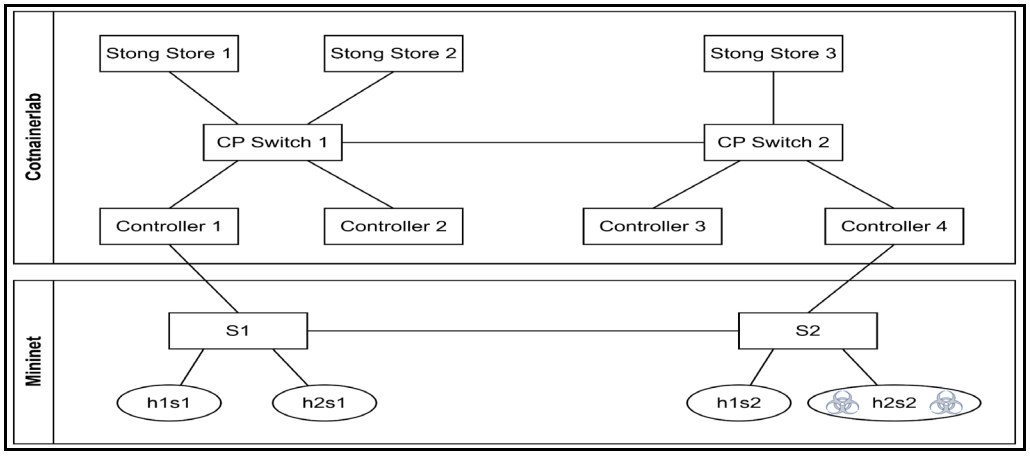

Figure 4: The topology used for use case testing

Table 1: Use case experiment

Step	Action	ONOS	PoC + HighAvail
1	Allow all devices	SUCCESS	SUCCESS
2	PingTest	Results: 0% dropped (12/12 received)	Results: 0% dropped (12/12 received)
3	Induce partition	SUCCESS	SUCCESS
4	PingTest	Results: 0% dropped (12/12 received)	Results: 0% dropped (12/12 received)
5	Deny h2s2 on controller 3	Process hangs	SUCCESS
6	PingTest	Results: 0% dropped (12/12 received)	h1s1 -> h1s2 h2s1 X h1s2 -> h1s1 h2s1 X h2s1 -> h1s1 h1s2 X h2s2 -> X X X *** Results: 50% dropped (6/12 received)
7	Fix Partition	SUCCESS	SUCCESS
8	PingTest	Results: 0% dropped (12/12 received)	h1s1 -> h1s2 h2s1 X h1s2 -> h1s1 h2s1 X h2s1 -> h1s1 h1s2 X h2s2 -> X X X *** Results: 50% dropped (6/12 received)
9	Wait for convergence	N/A	10 Seconds waited
10	Compare Strong and Eventual stores	N/A	Stores match

The experiment evaluates a controller utilizing Sridut against the base ONOS controller by inducing a split-brain partition and attempting to block a malicious host (h2s2). After the partition is induced, ONOS is unable to process any new ACL requests. Figure 5 displays a small excerpt of terminal output of the ONOS controller continuously trying to read from the strong store. Conversely, the PoC's SAL Observer detects the leadership loss (Figure 6), allowing the application to fall back to the CRDT Registry and successfully install the deny rule (Figure 7) on switch S2. After the partition is healed, the nodes gossip about any CRDT changes, which get applied to the strong store by the leader. Each node of the minority side will gradually switch back to using the strong store when its local changes have been applied and the strong store matches its local CRDT state. Since every strong write also commits to the CRDT store, verifiable convergence is still guaranteed under new updates to both the minority and majority sides.

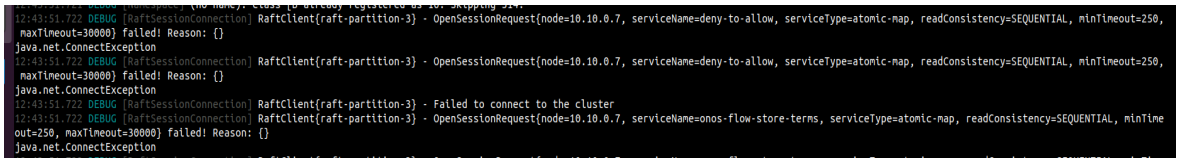

Figure 5: ONOS failing to connect to strong store

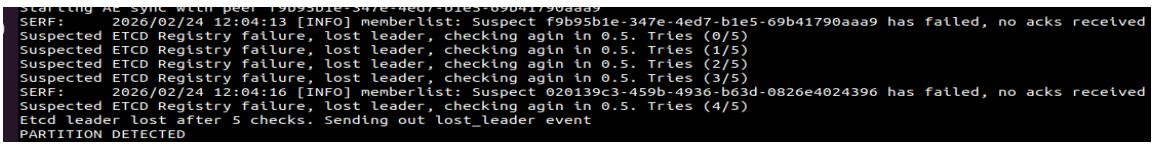

Figure 6: PoC detecting partition

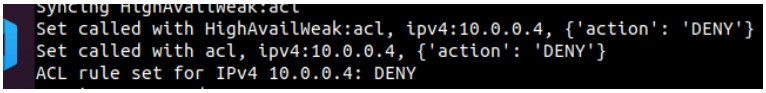

Figure 7: Deny rule being successfully installed

4.4 Discussion

The experiment shows that during a partition, the minority side of ONOS is essentially crippled and cannot make any progress. This is not a bug in ONOS, instead it is the intended functionality of a controller that treats the CAP theorem as a binary problem, where the minority can make no progress during a partition. In this case, however, it prevents the denial rule of the malicious host h2s2 from being installed on switch s2. Naturally, there are cases where applications should never make use of weaker consistency stores, as it could lead to further vulnerabilities (Xu, 2017; Yusuf et al, 2023). However, many controllers are architecturally restrictive (Ahmad and Mir, 2021), embedding consistency trade-offs directly into the controller. This leaves network administrators unable to align the control plane consistency with their specific risk profile. Sridut mitigates these risks by providing multiple stores and a unified API. This abstraction makes the selection between strong, eventual, or custom stores trivial at the application layer.

Regardless of the consistency model chosen, state transparency is critical. For example, in Table 1, the process hangs and times out as ONOS continuously tries to connect to its strong store. However, in a multi-threaded environment, this could lead to recursive retry loops for failed writes that could escalate into resource exhaustion (Munkhondya et al, 2023), creating an availability risk and a potential DoS vector. With Sridut, it is possible to query the state health as well as receive updates when services like the strong store fail. This allows services to react accordingly, such as keeping updates in a queue or returning useful error messages. While the use case experiment used the CRDT store in a supporting role, CRDTs can also be used as a replacement for some weak consistency stores. CRDTs are mathematically designed to operate without a central leader or synchronous consensus. This allows CRDTs to excel and operate securely in environments where network stability is not guaranteed (Sakic and Kellerer, 2018). In the use case experiment listed in Table 1, CRDTs are used to ensure that all nodes safely converge on the same network view. While this work has utilized CRDTs in an SDN environment, it has not fully explored all the possible security benefits. Instead, it serves as a basis for further research into how CRDTs can be applied across the broader SDN landscape to improve the control plane security.

5. Conclusion and Future Work

Current SDN implementations often treat the CAP theorem as a binary choice, forcing a full sacrifice of either integrity or availability during a partition. Additionally current controllers lack weak consistency stores with verifiable convergence, leading to integrity issues. To address this, this paper introduced Sridut, a secure multi-consistency storage model for SDN multi-controller networks. Sridut uses consistency state awareness to grant applications transparency to the health and state of the various consistency stores. Additionally, Sridut makes use of CRDTs to ensure verifiable convergence under any network conditions. Therefore, Stridut helps avoid critical integrity violations through verifiable convergence, and grants applications the ability to react in real time to consistency-based attacks. To evaluate Sridut, the authors developed HighAvail, a proxy application that leverages the state-aware primitives of the model to provide a multi-consistency store with verifiable convergence during network partitions. This application was tested against ONOS, demonstrating that while traditional controllers fail to enforce security policies during a partition, Sridut can enable the handling of security policies while still maintaining integrity. For future work, more research can be done to formally model the intricacies of utilizing CRDT and strong stores in tandem. Additionally, more resource optimization would be needed when utilizing multiple stores together in a production environment.

Acknowledgements

This research was supported by the Council for Scientific and Industrial Research (CSIR) through a postgraduate bursary

Ethics declaration: No ethical clearance was required for this paper.

AI declaration: Google Gemini was used as a tool for fact-checking technical citations and proofreading the paper for grammatical clarity. All other aspects of the paper, including the Sridut model itself were created manually by the authors.

References

Ahmad, S. and Mir, A.H. (2021) "Scalability, Consistency, Reliability and Security in SDN Controllers: A Survey of Diverse SDN Controllers," *Journal of Network and Systems Management*, 29(1), p. 9. Available at: https://doi.org/10.1007/s10922-020-09575-4.

Almeida, P.S., Shoker, A. and Baquero, C. (2015) "Efficient State-Based CRDTs by Delta-Mutation," in A. Bouajjani and H. Fauconnier (eds.) *Networked Systems*. Cham: Springer International Publishing (Lecture Notes in Computer Science), pp. 62–76. Available at: https://doi.org/10.1007/978-3-319-26850-7_5.

Bannour, F., Souihi, S. and Mellouk, A. (2020) "Adaptive distributed SDN controllers: Application to Content-Centric Delivery Networks," *Future Generation Computer Systems*, 113, pp. 78–93. Available at: https://doi.org/10.1016/j.future.2020.05.032.

Berde, P. *et al.* (2014) "ONOS: towards an open, distributed SDN OS," *Proceedings of the third workshop on Hot topics in software defined networking*. *SIGCOMM'14: ACM SIGCOMM 2014 Conference*, Chicago Illinois USA: ACM, pp. 1–6. Available at: https://doi.org/10.1145/2620728.2620744.

Brewer, E. (2012) "CAP twelve years later: How the 'rules' have changed," *Computer*, 45(2), pp. 23–29. Available at: https://doi.org/10.1109/MC.2012.37.

Brewer, E.A. (2000) "Towards robust distributed systems," *Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing*. *PODC00: ACM Symposium on Principles of Distributed Computing*, Portland Oregon USA: ACM, p. 7. Available at: https://doi.org/10.1145/343477.343502.

CNCF (2025) *etcd*, *etcd*. Available at: https://etcd.io/ (Accessed: January 20, 2025).

Das, A., Gupta, I. and Motivala, A. (2002) "SWIM: scalable weakly-consistent infection-style process group membership protocol," *Proceedings International Conference on Dependable Systems and Networks*. *International Conference on Dependable Systems and Networks*, pp. 303–312. Available at: https://doi.org/10.1109/DSN.2002.1028914.

David Brochart (2025) "pycrdt: Python bindings for Yrs." Available at: https://github.com/y-crdt/pycrdt (Accessed: June 16, 2025).

HashiCorp (2025) "hashicorp/serf." HashiCorp. Available at: https://github.com/hashicorp/serf (Accessed: March 10, 2025).

Hoang, N.-T. *et al.* (2022) "A Novel Adaptive East–West Interface for a Heterogeneous and Distributed SDN Network," *Electronics*, 11(7), p. 975. Available at: https://doi.org/10.3390/electronics11070975.

Kim, J. *et al.* (2024) "Ambusher: Exploring the Security of Distributed SDN Controllers Through Protocol State Fuzzing," *IEEE Transactions on Information Forensics and Security*, 19, pp. 6264–6279. Available at: https://doi.org/10.1109/TIFS.2024.3402967.

Lantz, B., Heller, B. and McKeown, N. (2010) "A network in a laptop: rapid prototyping for software-defined networks," *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. New York, NY, USA: Association for Computing Machinery (Hotnets-IX), pp. 1–6. Available at: https://doi.org/10.1145/1868447.1868466.

Marcel Rieger (2025) "pymitter: Python port of the extended Node.js EventEmitter 2 approach providing namespaces, wildcards and TTL." Available at: https://github.com/riga/pymitter (Accessed: May 17, 2025).

Munkhondya, H. *et al.* (2023) "Understanding Issues and Challenges of DFR Implementation in SDN Platform," *Procedia Computer Science*, 219, pp. 286–293. Available at: https://doi.org/10.1016/j.procs.2023.01.292.

Ongaro, D. and Ousterhout, J. (2014) "In Search of an Understandable Consensus Algorithm." *2014 USENIX Annual Technical Conference*, pp. 305--319.

openstack (2024) *os-ken*, *OpenDev: Free Software Needs Free Tools*. Available at: https://opendev.org/openstack/os-ken (Accessed: February 10, 2025).

Sakic, E. and Kellerer, W. (2018) "Impact of Adaptive Consistency on Distributed SDN Applications: An Empirical Study," *IEEE Journal on Selected Areas in Communications*, 36(12), pp. 2702–2715. Available at: https://doi.org/10.1109/JSAC.2018.2871309.

Shapiro, M. *et al.* (2011) "Conflict-free Replicated Data Types." *In Symposium on Self-Stabilizing Systems*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 386–400.

Smyth, D. *et al.* (2019) "Attacking distributed software-defined networks by leveraging network state consistency," *Computer Networks*, 156, pp. 9–19. Available at: https://doi.org/10.1016/j.comnet.2019.02.020.

Srl-Labs (2025) *containerlab*. Available at: https://containerlab.dev/ (Accessed: May 5, 2025).

Xu, L. *et al.* (2017) "Attacking the Brain: Races in the SDN Control Plane," *26th USENIX Security Symposium (USENIX Security 17)*.

Yusuf, M.N. *et al.* (2023) "Distributed Controller Placement in Software-Defined Networks with Consistency and Interoperability Problems," *Journal of Electrical and Computer Engineering*, 2023(1), p. 6466996. Available at: https://doi.org/10.1155/2023/6466996.

Zhang, Yuan *et al.* (2018) "A survey on software defined networking with multiple controllers," *Journal of Network and Computer Applications*, 103, pp. 101–118. Available at: https://doi.org/10.1016/j.jnca.2017.11.015.