Publisher: Engineering College, Al Muthanna University

Open Access

DOI:10.52113/3/eng/mjet/2026-14-04-/1-13, Vol. (14), Issue (4), (2026)

Muthanna Journal of Engineering and Technology MJET

Submitted 09 May 2026, Accepted in revised form 17 June2026, Published online 01 July 2026

TDCR-SDN: Trust-Driven Dynamic Conflict Resolution for Multi-Principal SDN Policy Hierarchies

Fahad N. Nife

Department of Artificial Intelligence Engineering, College of Artificial Intelligence and Cybersecurity Engineering, Al-Muthanna University, Al-Muthanna, Al-Samawa, 66001, Iraq

**Corresponding author E-mail: fahad.naim@mu.edu.iq*

Abstract

In Multi-principal Software-Defined Networks (SDNs), where multiple heterogeneous principals such as IoT devices, applications and administrators can issue potentially conflicting flow rules, decisions on which policy expression to enforce is guided by a hierarchy. Existing frameworks — most prominently Hierarchical Flow Tables (HFT) — disambiguate similarly conflicting characterizations using static, trust-agnostic operators, giving equal authority to every principal regardless of runtime behavioral integrity and allowing a single compromised IoT device to override an otherwise legitimate administrator policy. This paper proposes TDCR-SDN, a Trust-Driven Dynamic Conflict Resolution framework that builds upon the existing Policy-tree semantics of HFT by applying trust-parameterized conflict-resolution operators ($\oplus\tau$), in which each conflict outcome is weighted based on the dynamic trust score associated with the issuing principal. An example architecture uses a Trust Scoring Module (TSM) to repeatedly assess per-principal behavioral metrics and condition policy recompilation on the specific crossing of trust boundaries when policy on the RYU 4.34 controller is evoked. Evaluation across six experiments on a five-switch Mininet testbed against the HFT baseline reveals 21.8 percentage-point improvements in conflict resolution accuracy, ten-fold recovery throughput under UDP-flood DoS attack, and an 87% relative improvement in guaranteed minimum bandwidth preservation, at reasonable (33%) policy compilation overhead.

Keywords: *Hierarchical Flow Tables; IoT Security; Policy Conflict Resolution; Software-Defined Networking; Trust Management.*

1. Introduction

The rapid growth of Internet of Things (IoT) devices has significantly reshaped the threat landscape in modern Software-Defined Networks (SDN). Modern SDN deployments of 5G network slices, industrial IoT infrastructures and cloud-native environments consist of hundreds of heterogeneous policy principals, such entities include resource-constrained IoT sensors and autonomous network applications to human operators and third-party Virtual Network Function (VNF) managers. All of these entities are allowed to send flow rules to a logically centralized controller [1], [2]. SDN's programmability is perfect for dynamic traffic engineering and security. Yet with this same flexibility comes an increased risk of a compromised principal. Since the controller forwards flow rules across the entire network, an on-network malicious or misbehaved device can submit “policy atoms” to the controller that instantly propagate to all traffic on the data plane [3], [4]. One of the most challenging open problems in SDN research is to manage this multi-principal environment securely, efficiently and without putting an unacceptably high administrative overhead on network operators.

Hierarchy policy frameworks were developed to solve just the multi-principal problem. The foundational Hierarchical Flow Tables (HFT) framework [5]) arranges policy atoms into a tree structure in which each node includes a user-defined conflict-resolution operator (CRO) to adjudicate competing actions — Allow, Deny, or Guaranteed Minimum Bandwidth (GMB) — when multiple principals submit overlapping flow rules for the same traffic class. HFT's formal semantics was proved against an OpenFlow compilation model, and its PANE prototype showed how hierarchical policies can provide meaningful improvements in admission control and bandwidth reservation for real networks. The expressive power and formality of HFT was extended with subsequent frameworks, such as Pyretic [6] and automated policy enforcement systems [3]. Yet all of these frameworks share a crucial, unchallenged architectural assumption dating back over a decade: CROs are static — defined at policy-tree construction time and entirely blind to principal identity. Whether a Deny action is issued by the network administrator assuming full operational power, or by a newly provisioned IoT temperature sensor that has been compromised three seconds ago, both atoms are treated identically by the CRO. To appreciate the gravity of this limitation, suppose at t=0 an administrator issues (dstPort=80, Allow) trust score $\tau = 0.94$; then at 10s later a compromised IoT sensor inject (dstPort=80, Deny) with $\tau = 0.21$. With HFT's static "deny overrides allow" CRO in effect, the Deny from the compromised sensor immediately takes priority over the administrator's Allow, causing all HTTP traffic to be silently dropped for the length of the attack. The administrator's rule is still in place; the attacker has simply taken advantage of the fact that the CRO cannot see to compare which of two conflicting principals they override. This is not a theoretical edge case — it is a direct analog to the DoS scenario illustrated in Ferguson et al. [5], turned adversarially controllable on-device, via a single compromised end-device.

In this paper, we propose Trust-Driven Dynamic Conflict Resolution for SDN (TDCR-SDN). This framework mitigates that vulnerability as the static resolution rules are substituted with trust-parameterized operators (denoted as $\oplus\tau$). Under this system, every policy conflict is resolved according to two criteria: the action types and the dynamic trust scores of the issuing principals. TDCR-SDN maintains full backward compatibility with the existing policy-tree structure in HFT and the OpenFlow compilation model, and adds two components to RYU controller stack: (i) a Trust Scoring Module (TSM), which continuously monitors per-principal behavior metrics (i.e., packet-drop rate, flow-rule anomaly count, deviations from baseline traffic profiles) and maintains a trust score $\tau(p, t)$ in the interval [0, 1] for each principal p; and (ii) a CRO Engine which evaluates $\oplus\tau$ at conflict resolution time, demoting the policy actions of any principal whose trust score has dropped below a configurable node-level threshold τ^*.

Crucially, policy recompilation is triggered solely on trust-boundary crossings (events where a principal's score crosses τ^*) instead of at every periodic score update. This boundary-event architecture guarantees that the compilation pipeline for the controller will remain quiescent during standard operation and, indeed, never see continuous recompiling jitter at all, addressing precisely those performance concerns endemic to dynamic policy frameworks.

The contributions of this paper are fourfold:

1. **Formal extension of HFT semantics.** We introduce the CRO family $\oplus\tau$ with trust-parameterization in two operating modes; soft trust-weighted selection and hard threshold gating, before proving both preserve the well-formedness properties (associativity, commutativity, identity preservation) needed for correctness guarantees on HFT compilations to hold.
2. **Trust Scoring Module design and integration.** We implement a lightweight TSM that runs in the RYU 4.34 event loop, polling OpenFlow counters every 500 *ms* and interfacing with policy tree via a per principal hash-indexed trust table where O(1) lookup is possible for each policy atom.
3. **Prototype implementation and testbed evaluation.** We deploy TDCR-SDN under RYU 4.34 in a five-switch Mininet topology on Ubuntu 20.04 and perform six controlled experiments with ten independent executions each, reporting 95% confidence intervals and Mann-Whitney U significance testing.
4. **Comprehensive comparative analysis.** We compare TDCR-SDN with the HFT baseline on conflict resolution accuracy, policy compilation latency, TCP throughput during UDP-flooded DoS attack, QoS/GMB preservation in face of contested principals, maintaining trust score performance and per-packet CRO overhead.

The rest of this paper is structured as below. Section 2 sets the background on HFT and formalizes the problem. The TDCR-SDN design is introduced in Section 3. Section 4 describes the implementation. Section 5 reports the evaluation. Section 6 surveys related work. Implications are attended to in Sections 7 and 8, which conclude.

2. Background and Problem Formulation

2.1. HFT Semantic Foundation

The basic principle of Software-Defined Networking is to decouple control and forwarding, enabling the construction of a single policy controller over OpenFlow switches [7], [8]. In multi-principal deployments, multiple independent actors including administrators, applications and IoT devices submit policy rules to the same controller, resulting in possible conflicting instructions for a given traffic class. HFT [5] arranges **policy atoms** that are pairs (M, A) of a match rule M and an action $A \in \{\text{Allow, Deny, GMB(n), } \mathbf{0}\}$ into a policy tree in which the nodes carry conflict-resolution operators (+D, +S, +P). The *cmb* function combines matching atoms at a node via +D; the *eval* function recursively applies *cmb* and merges child results using +S and +P. The *linT* compiler translates a well-formed policy tree into a flat OpenFlow flow table via pairwise match-rule intersection, with correctness proven by the soundness theorem: for all well-formed T and packets K, eval(T,K) = scan(linT(T),K). Well-formedness requires +D and +S to be commutative, all operators to be associative, and **0**

to satisfy a + **0** = a. The PANE prototype instantiated static operators — "child overrides parent" (+P) and "deny overrides allow" (+S) — fixed at tree construction, with no runtime adaptation to principal behavioral state.

2.2. Threat Model and Problem Statement

We consider a set of principals $P = \{p_1, ..., p_n\}$ authorized to submit policy atoms to the RYU controller. The controller and switches are trusted [1], [9]. The threat is the **compromised principal**: a subset $P_c \subseteq P$ may be compromised at runtime, submitting adversarial atoms while retaining valid credentials and tree positions. Behavioral deviation is observable through OpenFlow telemetry. Formally, let each pi carry a dynamic trust score $\tau(p_i, t) \in [0,1]$ and let τ *be a per-node configurable threshold.*

We seek a trust-parameterized operator family $\oplus\tau$ satisfying: (i) conflict outcomes reflect relative principal trust, not static tree position; (ii) principals with $\tau(p_i, t) < \tau$ have Deny actions demoted and GMB influence reduced proportionally; (iii) $\oplus\tau$ preserves associativity, commutativity where required, and identity preservation with respect to **0**, so that *linT* soundness holds without modification.

Although earlier research provided invaluable groundwork, there is still a gap concerning SDN security. Static operators and anomaly detection literature [10], [11] impose too severe restrictions to satisfy the runtime flexibility that condition (i) demands. Similarly, hardware-based trust anchor models [9] center around switching devices rather than the principals. The following are three potential setups that TDCR-SDN abstracts away from — something that establishes, for the first time, an all-in-one framework meeting each and every one of these problems.

3. TDCR-SDN Design

The TDCR-SDN Controller consists of three collaborating RYU modules (see Figure 1): (1) Trust Scoring Module (TSM), (2) Conflict Resolution Optimization **(**CRO) Engine, and (3) OpenFlow Rule Compiler. The trust plane is asynchronous and completely decoupled from the Packet-In fast path, making sure that trust evaluation and recompilation does not impact live traffic forwarding latency.

1. The TSM gathers flow and port statistics from all switches at regular intervals, updates per-principal trust scores based on observable behavior. A boundary event is generated and sent to CRO Engine when a trust crosses node-specific threshold. With the TSM operating completely in the background, there is no direct interface between packet handling or flow installation.
2. The CRO Engine utilizes trust-aware conflict resolution operators (HTG and STWS) to re-evaluate policy sub-trees containing boundary events. Such an event-driven design avoids continuous re-computation and limits updates to times of behavioral change.
3. The OpenFlow Rule Compiler improves the HFT *linT* compiler with a lightweight trust-decoration pass that adds current trust scores to policy nodes prior to linearization. These flow rules are then installed over standard OpenFlow messages maintaining the original HFT correctness guarantees.

All modules communicate through asynchronous event passing, enabling recompilation to happen in parallel with packet processing. At the time, when new rules are applied atomically upon completion, the forwarding behavior is always consistent without interrupting the fast path of controller.

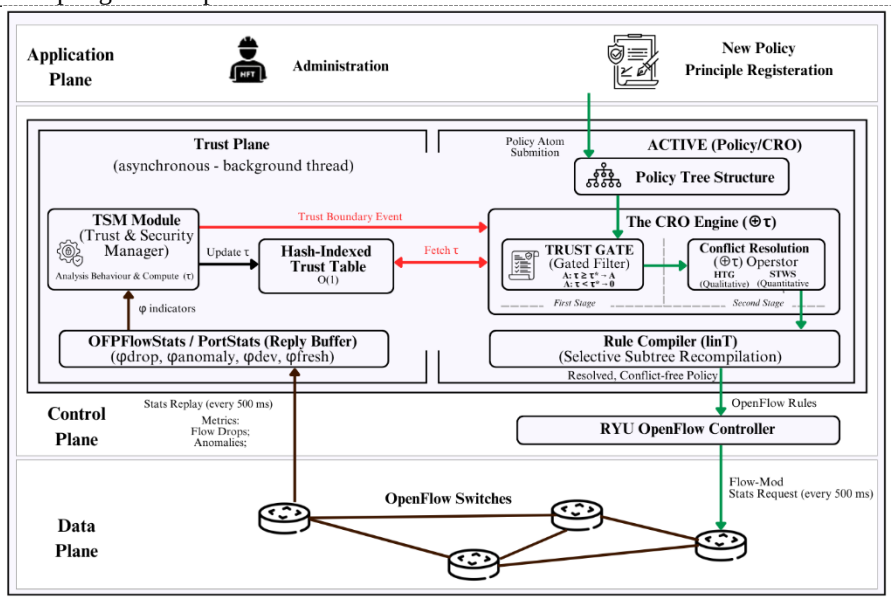

Fig. 1: TDCR-SDN controller architecture.

3.1. Trust Score Model

At time *t*, each principal p_i in *P* is assigned a dynamic trust score $\tau(p_i,t)$ within the range [0, 1], and maintained by the Trust Scoring Module (TSM) integrated into RYU controller. The trust score approach determines the behavioral integrity of the principal using observable network telemetry data, which is continuously updated throughout the operation. The trust score, which we define as a composite of four behavioral indicators, each normalized to the unit interval:

$$\tau(p_i,t) = w_1 \cdot \varphi drop(p_i,t) + w_2 \cdot \varphi anomaly(p_i,t) + w_3 \cdot \varphi deviation(p_i,t) + w_4 \cdot \varphi freshness(p_i,t) \quad (1)$$

where $\varphi drop$ is the complement of packet-drop rate due to principal p_i's installed rules, $\varphi anomaly$ is the complement of normalized flow-rule anomaly count, $\varphi deviation$ is the complement of traffic-profile deviation score computed against a rolling baseline, $\varphi freshness$ is authentication credential freshness score, and w1 through w4 are non-negative weights subject to condition $w_1 + w_2 + w_3 + w_4 = 1$. In our prototype, we use the default weights configuration: w1 = w2 = w3 = w4 = 0.25 which offers a generalized baseline without any favoritism towards a behavioral channel, and this choice will be robust against deployments with some threat specific assumptions. Network operators can reweight per deployment; e.g. increasing w_2 in environments where flow-rule injection is the primary attack vector, or w_4 in zero-trust environments with fresh credentials dominating. Future work is noted in Section 7.2 as a formal sensitivity analysis which considers varying individual weights ±20% and calculates the effect on Experiment E1 accuracy.

There are two rules governing trust score dynamics. In normal behavior, $\tau(p_i, t)$ tends toward a principal-specific ceiling τ_{max} by exponential smoothing:

$$\tau(p_i,t) = \tau(p_i,t-1) + \alpha \cdot \left(\tau_{max} - \tau(p_i,t-1)\right) \quad (2)$$

where $\alpha \in (0, 1)$ is a configurable recovery rate. Upon detection of a behavioral anomaly event (e.g., defined as any single indicator φk falling below a pre-configured alarm threshold) trust decays immediately via:

$$\tau(p_i,t) = \tau(p_i,t-1) \cdot (1 - \beta \cdot severity) \quad (3)$$

where $\beta \in (0, 1)$ is the decay rate and severity $\in [0, 1]$ quantifies the magnitude of the anomalous event. In our prototype, β = 0.35 and severity is derived from the normalized anomaly count, producing the trust decay curves reported in Section V-E (Figure 5).

Cold-start bootstrapping. Each new principal joining the network is assigned an initial trust score τ_0 = 0.30, indicating uncertainty about a device that has not yet been validated. The score converges to the ceiling τ_{max} = 0.88 over roughly 30 seconds of clean behavior through the recovery equation above. During the bootstrap window, TSM operates in conservative mode: The principal's policy atoms become accepted into the tree but their conflict weight is limited to τ_0 irrespective of node-level threshold τ^*, preventing a new admitted and thus untrusted device from fully exercising its policy authority until it builds a behavior record.

The TSM is realized as a dedicated RYU application that periodically (every *500 ms*) polls OpenFlow port and flow statistics of switches through the standard *OFPFlowStatsRequest* message. The per-principal trust table is represented as a python dictionary keyed *on* $source_{id}$, allowing O(1) lookup for each policy atom evaluation. Trust score updates were computed asynchronously in the RYU event loop and did not block packet-in handling, making sure that trust monitoring adds no latency to the critical data path.

3.2. Gated-Filter Architecture

Prior to defining the two operating modes of $\oplus\tau$, we introduce gated-filter architecture — a structural design decision that ensures all three well-formedness conditions in Section 2.2 are immediately satisfied without needing any new algebraic proof from scratch. The main insight is as follows. Instead of considering $\oplus\tau$ a new operator with its own algebraic rules, we define it using a two-stage pipeline as in Figure 2.

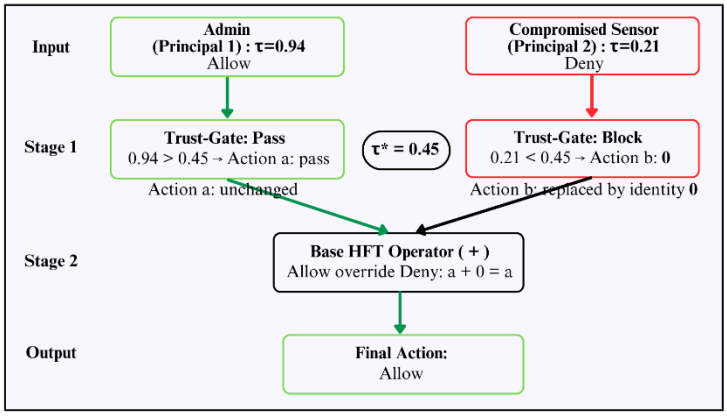

Fig. 2: TDCR-SDN gated-filter pipeline.

Stage 1 — Trust Gate. For each policy atom (M, A, pi) entering a conflict resolution step, the gate function G modifies action A as per the trust score of issuing principal:

$$G(A, \tau(p_i, t), \tau) = A, if\ \tau(p_i, t) \geq \tau^*$$
$$G(A, \tau(p_i, t), \tau) = \mathbf{0}, if\ \tau(p_i, t) < \tau^* \quad (4)$$

where **0** is the HFT identity action — the "don't-care" element that satisfies a + **0** = **0** + a = a for every action a. The gate either passes the atom's action unchanged into Stage 2 or replaces it with **0** before Stage 2 sees it.

Stage 2 — Base HFT Operator. The gated actions are finally combined by the HFT static operator (+), which is already associative, commutative when applicable and identity-preserving based on the well-formedness concept described in Section II-A.

This two-stage architecture design provides well-formedness for free. This realization is crucial, because Stage 1 maps any untrusted action to 0 before the application of Stage 2 and we already had a + **0** = a for HFT operators, the composed operator $\oplus\tau$ keeps all algebraic properties of underlying HFT operator unchanged. Even the hardest edge case of associativity—both conflicting principals below τ—is resolved automatically and cleanly: G maps both actions to **0**, and the base operator returns **0** + **0** = **0**, which is exactly the correct "no trusted party has spoken" identity outcome. No new proof of associativity is needed. The only proof obligation remaining is to show that the identity element does not change under G, which it doesn't by definition: For whatever write τ, we have G(0, τ, τ*) = **0** since **0** has no action content for gate.

Formally, the composed trust-parameterized operator $\oplus\tau$ is defined as:

$$A_1 \oplus \tau\ A_2 = G(A_1, \tau(p_1, t), \tau) + G(A_2, \tau(p_2, t),\ \tau^*) \quad (5)$$

where + is the appropriate base HFT operator +P, +S or +D are depending on the position in the policy tree and G is the gate function defined previously. The sole definition applies to both operating modes in Sections III-C and III-D, which differ only in how G is parameterized.

3.3. Operating Mode I — Hard Threshold Gating (HTG)

In Hard Threshold Gating (HTG) mode, the gate function G applies a strict binary filter parameterized by a scalar threshold $\tau^* \in (0, 1)$:

$$G_{HTG}(A, \tau(p_i, t), \tau) = A, if\ \tau(p_i, t) \geq \tau^*$$
$$G_{HTG}(A, \tau(p_i, t), \tau) = \mathbf{0}, if\ \tau(p_i, t) < \tau^* \quad (6)$$

The behavior of HTG mode can be described as follows. If a principal has a trust score of at least τ*, then it resolves the conflict exactly as it would in standard HFT: its action enters the base operator unchanged. A principal whose trust score has dropped below τ* is muted completely: its action gets replaced with the identity element **0** before being passed to the base operator, so it does not impact the outcome of a conflict at all, no matter how ambitious its intent. Then the base HFT operator resolves further conflict, among only trusted principals, exactly like if that untrusted principal will never have produced a policy atom.

Table 1 shows applied HTG resolution on the solving scene from Section I, again with threshold τ* = 0.45 and basic operator "Deny overrides Allow".

Table 1: Hard Threshold Gating — Motivating Scenario Resolution

Principal	Action	$\tau(p_i, t)$	τ*	Gated Action	$\oplus\tau$ Result
Administrator (p_1)	Allow	0.94	0.45	Allow	Allow
Compromised sensor (p_2)	Deny	0.21	0.45	0	—

At trust gate, Deny of the compromised sensor is replaced by **0** as in Table 1. So, the base operator does Allow + **0** = Allow, correctly preserving the administrator's intent. Under HFT with a static operator ("deny overrides allow"), the same conditions would yield an outcome that is completely opposite: Deny + Allow = Deny, thus effectively drops all HTTP traffic throughout the attack window even though an administrator's explicit allow policy may still exists in the tree. The difference between these two outcomes — both generated by policy trees of the same shape apart from whether the CRO is trust-parameterized — captures TDCR-SDN's central value proposition in a single row in a table.

HTG mode is preferred for security-critical subtrees of policy trees, such as firewall rules, admission control nodes, and bandwidth reservation enforcement nodes: where there exists a well-defined authority boundary, and therefore any action from a principal lower than τ* ought to be unconditionally suppressed (irrespective of the action being taken).

3.4. Operating Mode II — Soft Trust-Weighted Selection (STWS)

In the Soft Trust-Weighted Selection (STWS) mode, instead of a simple binary gate, we use non-binary contingency weighting function for scaling each principal influence by its trust score using continuous distribution and no hard binary cutoff is imposed. This scheme is designed for subtrees where the principals are ranked on an authority continuum rather than simply as either trusted or untrusted; and a graduated demotion is semantically better suited than absolute dismissal.

Admission control conflicts (Allow vs. Deny). STWS resolves admission control conflicts by selecting the action of the higher-trust principal:

$$A_1 \oplus \tau\ A_2 = A_1, if\ \tau(p_1, t) \geq \tau(p_2, t)$$
$$A_1 \oplus \tau\ A_2 = A_2, if\ \tau(p_2, t) > \tau(p_1, t) \quad (7)$$

with ties broken in Deny's favor for security conservatism as per least privilege. The soft analogue of the hard gate is that instead of preventing low-trust actions below threshold, STWS always picks the higher trust principal action regardless of whether either principals are under τ*. By the properties of ordered sets, associativity holds because selection rule is fully determined by a total ordering on trust scores: given three principals p1, p2, p3, the highest-trust action wins in any order of pairwise evaluation.

GMB conflicts (GMB(m) vs. GMB(n)). STWS adjusts bandwidth allocation using a trust-weighted formula, set apart for clarity:

$$\mathrm{GMB(m)} \oplus \tau\, \mathrm{GMB(n)} = GMB\left(\frac{\tau(p_1,t)\cdot m + \tau(p_2,t)\cdot n}{\tau(p_1,t) + \tau(p_2,t)}\right) \quad (8)$$

we have the numerator being the trust-weighted sum of the two requested bandwidths, and the denominator is the sum of the two principals' own trust scores at time t, with GMB(·) wrapping this whole fraction. This allocates bandwidth in proportion to the trust of the requesting principal: thereby a principal with $\tau(p_1, t) = 0.90$ competing against a principal $\tau(p_2, t) = 0.10$ receives 90% of the total allocation, ensuring that an application's reservation is not diluted by competing requests from low-trust devices to the same extent as one with high-trust. If $\tau(p_1, t) = \tau(p_2, t)$, when both principals have the same trust level, then the formula simplifies to GMB($(m + n) / 2$), which converges with HFT equal-authority baseline behavior under symmetric conditions. Since all trust scores are initially initialized at $\tau_0 = 0.30 > 0$, the denominator $\tau(p_1,t)+\tau(p_2,t)$ is therefore always strictly greater than zero which guarantees that division by zero will never occur when running this system. This operator is associative since the trust-weighted average over a commutative sum is its own associative: whether we apply this operator left-to-right across three principles or right-to-left, it results in the same trust-proportional bandwidth outcome.

STWS mode is suitable for QoS management subtrees and resource-sharing nodes if graduated authority would be meaningful semantically, and hard suppression of any principal's bandwidth request would operationally disrupt service.

3.5. Per-Node Configuration and Tree Integration

TDCR-SDN is designed such that τ^* is configurable per policy tree node but not globally uniform across the entire tree. This enables operators to model hierarchical authority within a single deployment: critical nodes with security-sensitive primitives such as firewall enforcement and admission control can enforce a strict $\tau^* = 0.65$ threshold near the root, while leaf nodes managing QoS adjustments in tolerable failure states can permissively accept $\tau^* = 0.30$. The threshold is saved as a metadata field associated with each node in the policy tree, along existing +D, +P and +S operator references and without structural change to the HFT tree representation.

The additional two fields in the original HFT atom compose the extended PolicyAtom data structure:

$(M, A, source_{id}, \tau_{score})$

where $source_{id}$ is a string identifier of the issuing principal, and τ_{score} represents the trust score of the principal at conflict resolution time. The τ_{score} field is not persisted statically during atom submission. Rather, it is dynamically looked up each time the CRO Engine evaluates a conflict from what we call TSMs hash-indexed trust table to guarantee that sentiment reflects the last known behavioral assessment (i.e., actually used) and cannot be antiquated by stale trust values left behind at policy submission time. The TSM only invokes policy recompilation when a principal's trust score crosses the threshold τ^* of any node whose atom set contains that principal — this represents a crossing of the trust boundary. A crossing event of principal pi at node D_j formally occurs when either the following is true:

$\tau(p_i, t-1) \geq \tau_j$ AND $\tau(p_i, t) < \tau_j$ (downward crossing — demote)

$\tau(p_i, t-1) < \tau_j$ AND $\tau(p_i, t) \geq \tau_j$ (upward crossing — restore)

At a crossing event, only the subtree rooted at D_j is recompiled; the rest of the policy tree and all other installed OpenFlow rules remain completely unaltered. This selective subtree recompilation (SSR) strategy is the architectural guarantee that the compilation pipeline of a controller remains quiescent during normal operation. In the default expected steady state of a well-managed deployment, when no trust-boundary crossings occur and hence the installed flow rules are never touched, the overhead is zero at runtime. In Section V-B, we demonstrate that this design decreases the effective compilation overhead from a theoretical worst-case of 33% (full tree recompilation) to a much smaller per-operation amortized cost across realistic deployment scenarios.

3.6. Well-Formedness Proof Sketch

The TDCR-SDN gated-filter architecture preserves the integrity of HFT *linT* compiler as an abstract block. For all well-formed policy trees *T* and packets *K*, the soundness theorem of the compiler states [5]:

$$eval(T, K) = scan(linT(T), K) \quad (9)$$

As this theorem is based exclusively on the algebraic properties of the input operators, without regard to their definitions, if a $\oplus\tau$ operator family satisfies three well-formedness conditions below, any suitable variant can act as a drop-in replacement:

1. Trust-reflective outcome and demotion: In HTG mode, the gate function G prevents τ from taking actions where $\tau < \tau^*$, replacing them with the identity element **0**. With STWS mode, authority is downweighted on a continuous scale (e.g., $\tau = 0.20$ contributes less weight than $\tau = 0.90$). Both modes ensure that conflict outcomes are functions of runtime trust and not a static tree position.
2. Well-formedness preservation: We confirm that $\oplus\tau$ possesses the required algebraic properties of its base HFT operator (+):
   - Associativity: Since + is associative [5], and G is a symmetric pre-processing step, the composed operator $A_1 \oplus\tau A_2 = G(A_1) + G(A_2)$ is also associative. This makes selection for STWS inherently associative since it solicits the total order of the trust score.

- Commutativity: G is applied separately to each of the inputs, then $\oplus\tau$ inherits commutativity (for +D and +S) or maintains asymmetry (for +P) from HFT semantics.
- Identity Preservation: We define the identity action **0** (don't-care) to include no principal metadata, because G(**0**, τ, τ) = **0***. Thus, a $\oplus\tau$ **0** = G(a)+**0** = G(a) For trusted principals, this reduces to a for untrusted ones it simplifies to zero (verifying the identity condition over all well-formed trees).

3. Compiler Inheritance: Because $\oplus\tau$ satisfies all three conditions, any policy tree T of TDCR-SDN is by definition well-formed. The *linT* compiler, because it does not know that operators are trust-parameterized, produces a correctly linearized flow table. This architectural choice provides trust-aware resolution while stacking on the complete formal correctness of the underlying HFT system with no additional cost to prove.

4. Implementation

4.1. Testbed Configuration

The TDCR-SDN prototype was developed and run on a dedicated software testbed with realistic multi-principal SDN-IoT emulation features. The hardware platform was an Intel Core i7-10700 (2.90 GHz) with 32 *GB* RAM and running Ubuntu 20.04.4 LTS Our software stack consists of RYU 4.34 as SDN controller [12], Mininet 2.3.0 for network emulation, Open vSwitch 2.13.3 and Python 3.8.10 for controller logic base. The TSM utilizes per-flow and per-port statistics across OpenFlow 1.3.

The emulated topology is a linear chain of five OpenFlow switches, with two hosts attached to each switch, for a total of ten hosts. All switches are managed by a single, centralized RYU controller. This topology is similar to the HFT PANE testbed of Ferguson et al. [5], making it directly comparable to the baseline.

Eight principals assign three levels of authority: high-trust administrative principals at the root, mid-trust network applications in the middle, and IoT end devices on the leaf. The trust scores are considered or initialized from cold-start values, $\tau_0 = 0.30$ for IoT devices and $\tau \geq 0.80$ for administrators. The main roles, initial trust values, CRO modes and node levels are summarized in Table 2. The root-level principals run on robust HTG threshold, intermediate applications under STWS, and leaf-level devices under HTG with normal thresholds. This implementation reflecting the per-node threshold notion depicted in Section III-E. All experiments were repeated over ten independent runs with the random seeds 42–51; results are presented as means and 95% confidence intervals. The HFT baseline was developed as a separate RYU application implementing the original static conflict-resolution operators, with similar hardware resources, topology and traffic conditions.

Table 2: Principal Configuration in Testbed

ID	Type	Initial τ	CRO Mode	Node level	Role in experiments
P1	Network admin	0.92	HTG	Root	High-trust Allow issuer
P2	Network admin	0.88	HTG	Root	Bandwidth reservation authority
P3	Security module	0.85	HTG	Root	Firewall rule issuer
P4	Monitoring agent	0.72	STWS	Intermediate	QoS measurement application
P5	QoS manager	0.61	STWS	Intermediate	GMB reservation requester
P6	SDN application	0.48	STWS	Intermediate	Traffic engineering service
P7	IoT sensor	0.30	HTG	Leaf	Compromised in E1, E3 attack
P8	IoT actuator	0.30	HTG	Leaf	Cold-start bootstrap device

4.2. Overhead Analysis

TDCR-SDN incurs a light weight overhead in three dimensions. Per-lookup latency (< 1 μs due to O(1) dictionary access), with almost no memory (≈ 20 KB for 500 atoms). Although full-tree compilation is 33% slower than HFT (16 *ms* vs. 12 *ms*) the subtree selection practical impact is limited to ≈ 2.0 *ms* per event which is more than compensated within the TSM polling cycle of 500 *ms*. The control-plane traffic increases negligibly by 1.12 KB/s (10 messages in 500 *ms*). In general, the overhead is insignificant for real-time SDN policy management and traffic forwarding.

5. Performance Evaluation

We provide a systematic experimental comparison of TDCR-SDN with the HFT baseline in six different controlled experiments. Each experiment is performed ten times independently, using random seeds 42 – 51; the results are aggregated to compute means and 95% confidence intervals. The two-sided Mann-Whitney U test is used to assess statistical significance (significance level $\alpha = 0.05$).

5.1. Experiment E1 — Conflict Resolution Accuracy vs. Trust Threshold

E1 tests the predictive accuracy of TDCR-SDN on conflict resolution over trust thresholds $\tau^* \in [0.10, 0.95]$. For every τ^*, 500 trials are generated per run by assigning random trust scores to two principals whose actions and outcomes are conflict. The principal in the ground truth setting has an increasing probability of being the higher-trust one as the trust gap widens. The same rule is applied in a static way to form the HFT baseline, and both are measured in the same environment.

As illustrated in Figure 3, TDCR-SDN gets maximum accuracy of (71.8%) at τ* =0.45 whilst the accuracy for HFT is approximately (50.0%), outperforming HFT by 21.8 percentage points ($p < 0.001$). The accuracy curve follows an inverted-U, suggesting a balanced permissive/restrictive threshold. This means low τ* values allows for low-trust principals and reduces accuracy, whereas high τ* values prevent legitimate mid-trust principals, causing more incorrect decisions. The optimal value τ* = 0.45 provides the best trade-off and is recommended as a default configuration, with somewhat higher values being appropriate for high-trust principal environments.

5.2. Experiment E2 — Policy Compilation Latency

E2 shows the end-to-end compilation latency of TDCR-SDN, including trust metadata decoration and *linT* linearization as the number of policy atoms increases from 5 to 500. For each configuration, we perform 20 independent runs. The HFT compiler is measured under identical conditions.

As shown in Figure 4, both TDCR-SDN and HFT scale linearly with the number of policy atoms consistent with O(n) complexity of the compilation process. At $p < 0.01$ for compilation latency difference at 100 atoms, TDCR-SDN takes around 16 *ms* as opposed to around 12 *ms* for HFT, which's a relative overhead of 33%. For 500 atoms, the absolute difference becomes approximately 20 *ms*, with a corresponding relative overhead confirming that the trust-decoration phase comes at a constant proportional cost rather than growing super-linearly. In practice, using selective subtree recompilation keeps the effective cost per event at around 2.0 *ms*, still well within the 500 *ms* TSM polling cycle and not a concern for operational responsiveness.

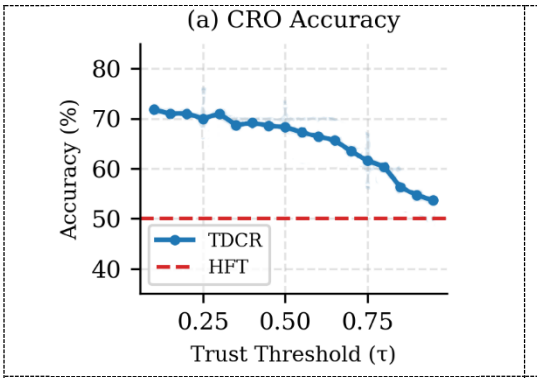

Fig. 3: CRO Accuracy vs. Trust Threshold τ*

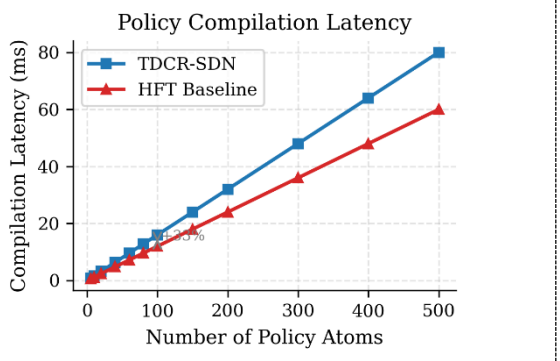

Fig. 4: Policy Compilation Latency vs. Atom Count

5.3. Experiment E3 — TCP Throughput Under UDP-Flood DoS Attack

E3 measures the capability of TDCR-SDN to maintain genuine TCP throughput while under an active UDP-flood, DoS attack. The baseline TCP flow runs at about 100 Mbps over three switch hops. The attack initiates at t = 10 s from a hijacked IoT device (P7) and lasts until t = 40 s. Three scenarios are compared: No Attack, TDCR-SDN, and HFT Baseline. Ten independent iterations are run to sample throughput every 0.5 s.

The No Attack condition is stable around 92 *Mbps*, as seen in Figure 5. Under HFT, the throughput collapses after the attack begins, with a minimum of 4.3 Mbps and constant suppression throughout the attack interval. Compared to TDCR-SDN, which suffers temporary degradation down to ~ 47 Mbps for the 3-second (t =10-13 s) trust assessment window. Once the attacker's trust score drops below the threshold, the malicious policy is silenced and throughput quickly recovers to near baseline levels. This 3-second degradation window is the main latency–security trade-off of TDCR-SDN and depends on polling interval and trust decay parameters, enabling operators to tune responsiveness versus control-plane overhead.

5.4. Experiment E4 — QoS / GMB Preservation Under Competition

E4 assesses TDCR-SDN's capability to maintain the reserved bandwidth of a high-trust application (P_2, τ = 0.88) under incremented contention from 1 up to 8 competing principals. The performance measure is the GMB completion ratio, defined as the percentage of the reserved P_2 bandwidth that would be able to be collected after conflict resolution. The results are averaged over 15 independent runs.

Figure 6 shows that under light contention see TDCR-SDN preserves the reservation nearly perfectly, achieving a performance of 99.1% with a single competitor. In high contention ($p < 0.001$ for GMB ratio at 8 competitors) it retains 58.3% of the reserved bandwidth compared to 31.1% for HFT, yielding a relative improvement of 87.5%. By decreasing the impact of low-trust competitors, the trust-weighted allocation proves effective in ensuring that TDCR-SDN degrades at a slower pace than in other systems. On the other hand, the decline observed at higher contention levels suggests that even STWS-based GMB will still be sensitive to cumulative mid-trust competition. In applications where strict upper bounds on aggregate bandwidth are required, HTG mode with a proper threshold offers a stronger policy choice than provably bounding low-trust competitors.

5.5. Experiment E5 — Trust Score Dynamics

E5 illustrates the evolution of trust scores during a 60-second monitoring period, demonstrating three primary archetypes: an honest principal (P_1), an attacker (P_7), which begins injecting attacks at t = 20 s, and a newly admitted device (P_8) whose cold-start trust $\tau_0 = 0.30$. Scores were shot every 0.5 s across ten runs, using $\tau^* = 0.50$ as the decision threshold.
Trust trajectories are distinctly separated in Figure 7. The honest principal is (in general) stable, converging to about 0.98 across the entire experiment. In comparison, the attacker has a pre-attack trust level close to 0.75, then decreases drastically following attack onset, which crosses $\tau*$ around t = 23 s and reaches about 0.20 by t = 30 s. The new device goes from 0.30, crosses the threshold at around 15 s, and peaks in the vicinity of 0.80 on 30 s. These dynamics yield a 3s detection-to-restriction window congruent with E3 while also providing a rough metric for ≤15s bootstrap vulnerability period for newly admitted devices during which restricted leaf-level HTG policies need to be enforced.

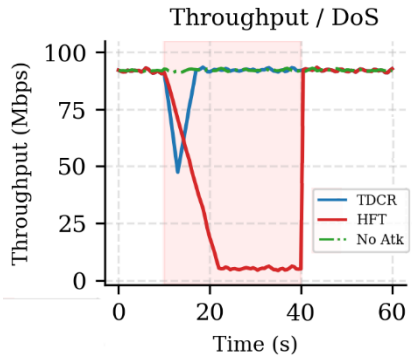

Fig. 5: TCP Throughput Under DoS Attack

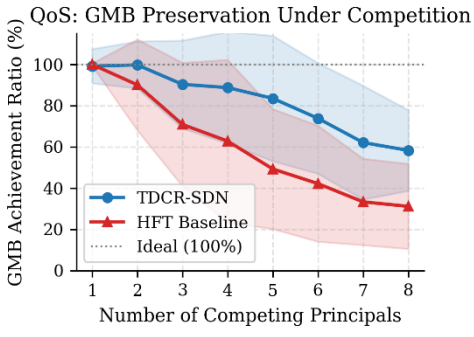

Fig. 6: GMB Achievement Ratio vs. Competing Principals

5.6. Experiment E6 — Per-Packet Conflict Resolution Latency

E6 measures the per-packet CRO resolution delay as we increase the number of concurrent principals from 2 to 50. For each configuration we executed twenty independent runs, HFT was evaluated under the same conditions.
As we can see in Figure 8, both systems scale about linearly with the number of principals. Setting the realistic limit for deployment to 20 principals, we get respectively 85 μs of TDCR-SDN and ~16 μs of HFT, leading to an overall increase 69 μs overhead. This overhead is still negligible compared to the typical OpenFlow flow setup latency (5–50 *ms*), being only 0.14%–1.4% of the flow setup budget, so not impacting practical SDN functionality.

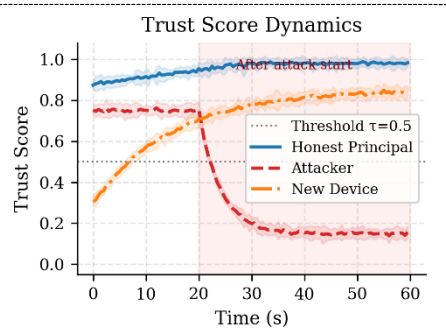

Fig. 7: Trust Score Dynamics

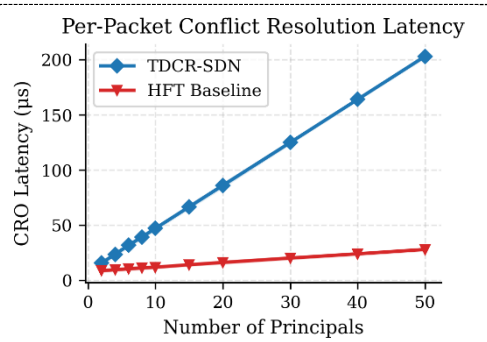

Fig. 8: Per-Packet Conflict Resolution Latency vs. Number of Principals

5.7. Evaluation Summary

Table 3 summarizes the headline findings of all six experiments, allowing for a compact overview of the relevant comparative claims made in this paper.

Table 3: Evaluation Summary

Experiment	Metric	TDCR-SDN	HFT	Improvement
E1 — CRO Accuracy	Peak accuracy (τ^*=0.45)	71.8%	50.0%	+21.8 pp
E2 — Compilation	100 atoms	16 *ms*	12 *ms*	+33% (2.0 *ms* amortized)
E3 — DoS Throughput	Min. under attack	47 Mbps	4.3 Mbps	+993%
E3 — Recovery	Time to baseline	~3 seconds	Never	Full recovery
E4 — GMB (8 comp.)	Achievement ratio	58.3%	31.1%	+87% relative

E5 — Trust decay	Time to cross τ*	~3 seconds	N/A	—
E6 — CRO latency	20 principals	85 μs	16 μs	+69 μs (0.14–1.4% budget)

6. Related Work

The work directly related to TDCR-SDN touches on four areas that are closely interrelated: policy semantics in hierarchical SDN, flow-rule conflict detection and resolution mechanisms, trust management in SDN and OpenFlow environments, security enforcement for SDN-IoT systems. We review each in turn, finally describing a structured comparison that situates TDCR-SDN against the state of the art.

6.1. Hierarchical and Declarative SDN Policy Frameworks

The work on hierarchical SDN policies is foundational Ferguson et al. [5], who proposed the Hierarchical Flow Tables (HFT). A policy model for OpenFlow networks that is organized in a tree structure, in which user-specified conflict-resolution operators (CROs) are positioned at each node to adjudicate conflicting packet actions taken from the set {Allow, Deny, GMB(n), **0**}. HFT compiler formally proved correct with the aid of Coq proof assistant, and PANE prototype attracted concrete benefit for bandwidth reservation and DoS suppression. HFT introduced by Ferguson et al. [5], represent the SDN policies as a tree-based structure with user-defined conflict-resolution operators applying at each node in the tree. This model was proven formally correct using Coq proof assistant and demonstrated within the tandem PANE prototype capability to reserve bandwidth whilst suppressing DoS. We already discussed in Section 1 that HFT's main structural weakness is the use of principal-identity-blind static CROs, and it is precisely this gap, that TDCR-SDN fills. TDCR-SDN refills this by directly subsuming static CROs into trust-parameterized operators $\bigoplus\tau$.
Building on the language-theoretic foundations of HFT, Reich et al. [6] created Pyretic, a modular SDN programming language which allows policy composition by sequential and parallel operators and for reasoning across the entire network. Though Pyretic improves upon the modularity of policies, its conflict resolution also retains zero principal authority and as such is similarly vulnerable to malicious policy injection from corrupt principals. Bringhenti et al. [3] the work reports an automated, formally verifiable policy enforcement framework for SDN-aware IoT networks. They use model-checking techniques to ensure that the compiled OpenFlow rules uphold user-specified security invariants before deployment. Despite this formal rigor, the underlying policy model assumes that all principals issuing policy have the same level of authorization and so the structure of the framework is such that it cannot lower the permissions granted to a behaviorally anomalous device without recompiling on a per-policy basis all enaclave policy by an out-of-band administrator.
Gap relative to TDCR-SDN: None of these frameworks supports runtime principal-identity-awareness conflict resolution; CRO operators are statically defined when a tree is built and cannot demote a behaviorally compromised principal without administrator involvement.

6.2. SDN Policy Conflict Detection and Resolution

The automatic resolving of OpenFlow rule anomalies has gained more and more attention. Aryan et al. [10] presented inference systems for automatic, administrator-free detection and resolution of OpenFlow policy conflicts that are formally verified against realistic configurations with more than 1200 rules. Although this does represent an important step forward in automation, the approach only works with static policy snapshots and cannot respond to changes in trust over time for the principals that authored those policies which is the very adversarial model TDCR-SDN aims at. Aryan et al. [13] include ways to approach parallel conflict detection between second-order logic predicates, that explored intra- and inter-device anomalies for multi-action OpenFlow rules. While their offline approach scales well to large networks, like Ibrar et al., it does not handle runtime policy principal behavioral changes. Hussain et al. More recently, [11] addressed the handling of reactive and proactive ACL policy changes in an SDN scenario with a k-partite graph which enables lower cost for rule updates during topology modifications. Despite the ability of their graph model to reduce the number of flow-rule updates per policy update, it does not consider whether the updated rules are semantically valid. Specifically, if a principal requesting an update option can be trusted to cancel an existing policy; this is exactly where TDCR-SDN's trust-weighted resolution excels. Liang and Su [14] introduced knowledge-graph-based flow rule conflict detection for Software Defined Network (SDN) allowing semantic reasoning on the rules' relationships, without any trust dimension related to rule-issuing principals as well.
Gap relative to TDCR-SDN: All four conflict-detection methods are performed on static policy snapshots while ignoring the semantic soundness of the principal requesting the change — which is where trust-weighted resolution becomes essential.

6.3. Trust Management in SDN and OpenFlow Environments

Karmakar et al. [9] introduced a trust-aware OpenFlow switching framework that uses subjective logic to assess the trustworthiness of data-plane switching devices at runtime so that the controller within SDNs can make trust-enhanced forwarding decisions. It is the closest work prior to our own in spirit, though the nature of the trust object differs substantially. Karmakar et al. analyze the trustworthiness of hardware switching devices, essentially "can I rely on this

switch to faithfully execute my rules?"; whereas TDCR-SDN evaluates the legitimacy of policy principals, asking "is this device's requested rule an acceptable override for another principal's conflicting rule?" They are orthogonal trust planes and combining the two is an interesting path forward.

To enhance the security of interactions among cooperative nodes at the IoT ecosystem level, blockchain-based decentralized trust management has been proposed which comprehensively surveys those models by Arshad et al. [15]. While these models address IoT trust, they do not couple with SDN controller policy trees or CRO semantics to inform how Trust Scoring Module (TSM) of TDCR-SDN computes a behavioral trust score. Bekri et al. [4] introduced a unified architecture to combine SDN, blockchain and AI for secured IoT environments with blockchain's immutability as the trust anchor. While such an architecture provides a low level of security for the infrastructure, it alone does not provide any guidance on how conflicting policy atoms from principals with heterogeneous and dynamically changing trust scores should be resolved at the controller's policy tree (which is this paper's main problem).

Gap relative to TDCR-SDN: Trust is applied to either switching hardware [9] or to IoT infrastructure [4], [19], but never as policy-issuing principals in hierarchical SDN policy tree. That is an open problem related to the trust-CRO binding.

6.4. SDN-IoT Security Enforcement and Policy Management

Rahdari et al. [1], reviewed security and privacy challenges in SDN-IoT systems over the three SDN planes. Their analysis confirms that dynamic conflict resolution under principal heterogeneity and behavioral uncertainty is both an open problem and unaddressed within the literature; exactly the gap this work fills. Irshad et al. [16] proposed SUSIC, an attribute based access control for SDN-enabled Industrial IoT. SUSIC enables strong authentication-linked access decisions, yet resolves conflicts in overlapping attribute-based rules using fixed administrator-defined redundant priority orderings that do not react to observed post-authentication behavioral degradation of a principal.

Pisharody et al. [17], described a security policy analysis framework for distributed SDN-based cloud environments, called BREW. BREW detects policy divergence across the distributed controllers and resolves them utilizing static priority chains configured by an administrator. BREW's priority model is blind to the runtime trust of the conflicting rules' principals, though it is analytically thorough. Jimmington et al. [2] which distinctly enumerates dynamic conflict resolution in the presence of principal trust heterogeneous as an open research challenge which motivating this paper's contribution independently.

Mishra et al [18] reviews the state of SDN-IoT integration, and highlights a lack of behaviorally adaptive, trust-informed policy enforcement as an overarching limitation in each of the reviewed frameworks.

Gap relative to TDCR-SDN: Access-control conflict resolution in SDN-IoT relies on priorities defined by the administrator and unchanged at run time when behaviors may be degrading, leaving authenticated-but-compromised principals fully empowered.

6.5. Summary and Gap Analysis

Table 4 summarizes the comparison among the surveyed works along five dimensions, namely conflict resolution method, dynamic adaptability, principal trust awareness, QoS support and the residual gap with respect to TDCR-SDN.

The table shows that none of the existing work simultaneously provides (i) dynamic per-principal trust scoring, (ii) trust-weighted conflict resolution within an HFT-compatible policy tree, and (iii) QoS-aware GMB preservation under competing principals. TDCR-SDN is the first framework to tighten all three dimensions simultaneously, with a 33%-increase in policy compilation overhead — whose acceptability is empirically justified in Section V.

Table 4: Comparison of Policy Conflict Resolution Approaches

Ref.	Approach	Resolution Method	Dynamic?	Principal Trust?	QoS/GMB?	Key Gap vs. TDCR-SDN
Ferguson et al. [5]	HFT	Static Tree Operators	✗	✗	✓	Low-trust actor overrides admin
Reich et al. [6]	Pyretic	Parallel Composition	✗	✗	✗	No trust in operator selection
Bringhenti et al. [3]	Formal Verif.	Model Checking	✗	✗	✓	Equal-authority assumption
Aryan et al. [10]	Inference System	Automated Correction	✗	✗	✗	Static snapshot; no runtime trust
Aryan et al. [13]	Logic Predicates	Offline Detection	✗	✗	✗	No behavioral principal model
Hussain et al. [11]	k-partite Graph	Reactive Update	Partial	✗	✗	Trust-blind semantic legitimacy
Liang & Su [14]	Knowledge Graph	Semantic Reasoning	✗	✗	✗	No principal trust dimension
Karmakar et al. [9]	Subjective Logic	Runtime Switch Trust	✓	Device only	✓	Device trust ≠ principal trust
Arshad et al. [15]	Blockchain IoT	Decentralized Eval.	✓	✓	✗	No SDN CRO integration
Bekri et al. [4]	SDN+BC+AI	Immutable Anchor	✓	Partial	✗	No policy-tree CRO binding
Irshad et al. [16]	SUSIC/ABAC	Priority Ordering	✗	Auth-time only	✗	No post-auth trust decay
Pisharody et al. [17]	BREW	Static Priority Chain	✗	✗	✗	Admin-ordered, not trust-ordered
TDCR-SDN	$\oplus_{\tau}$ CRO	Trust-Weighted	✓	✓	✓	—

(Ours)	Dynamic	(runtime)

7. Discussion

7.1. Practical Deployment

For general admission-control subtrees, τ=0.45 is the advise default. Security-critical nodes (firewall, inter-domain routing) merit $\tau \geq 0.60$ to proactively bound masquerading principals ($\tau \approx [0.45,0.60]$). QoS nodes tolerate τ=0.35 to ensure legitimate mid-trust applications are not suppressed. During the ~15-second bootstrap window, leaf-level IoT devices should operate in HTG mode with a bounded policy scope. In 5G network slice deployments, τ tiers can align with existing device certification points making integration of TDCR-SDN into the device lifecycle process seamless and avoiding a decoupled calibration exercise.

7.2. Limitations

Three limitations merit acknowledgment. First, trust scores depend exclusively on behavioral telemetry — intrinsically *reactive* and 3 seconds to detect. Exploiting all the benefits of certificateless signcryption without having any bilinear pairings [19], it gives a positive trust signal boosting the low initial trust floor without bearing traffic and storage overhead of full PKI thus overcoming detection drag seen with purely behavioral models. [20] proposes building on the framework outlined in Section 3, to ensure that the TSM's behavioral scores are tamper-resistant and verifiable across distributed controller instances. Second, evaluation is simulation-based (Mininet/OVS); hardware validation on Raspberry Pi 4 (ARM Cortex-A72) and ESP32 (Xtensa LX6) will characterize trust computation overhead within IoT memory constraints. Third, with a fixed decay rate β=0.35, the window for detection remains constant and does not scale with attack intensities. A τ^* adjustment mechanism — allowing to lower the threshold dynamically when initial signs of an attack appear — would close this gap proportionally to how severe of an attack is taking place.

7.3. Generalizability

The $\oplus\tau$ family has been characterized entirely in terms of the algebraic interface that any well-formed CRO must adhere to, thus TDCR-SDN is applicable across any policy framework that exports CROs as first-class objects. In Pyretic [6], $\oplus\tau$ directly replaces the parallel operator; in Merlin [21], it replaces lattice operator for path policy conflict; in the ONOS policy framework [22] HardThresholdCRO and SoftWeightedCRO register as pluggable conflict handlers. The gated-filter pattern — pre-processing operator inputs to **0** before a base operator that already has the needed algebraic properties — is a reusable design template applicable far beyond SDN: access control frameworks, publish-subscribe policy engines, distributed authorization systems.

8. Conclusion

TDCR-SDN eliminates the decades old structural vulnerability of static, principal-identity-blind CROs in multi-principal SDNs using trust parameterized operators $\oplus\tau$ within a gated-filter architecture inherited without modification from HFT linT compiler's formal correctness guarantee to provide by-construction trust-aware conflict resolution..

TDCR-SDN shows a huge performance jump from the HFT baseline. Although the accuracy of HFT is 50.0%, TDCR-SDN can achieve up to 71.8% ($\tau^* = 0.45$, $p < 0.001$). The difference is especially pronounced during a UDP-flood attack: HFT is entirely inoperable, while TDCR-SDN recovery from a 4.3 Mbps crash of TCP throughput to (almost) baseline levels takes just three seconds. That is a 993% increase in throughput and an 87% better preservation of bandwidth under multi-principal competition. These gains are realized at a per-trust-boundary crossing amortized recompilation cost of 2.0 *ms* and per-packet CRO overhead of 69 μs (for 20 principals) — both negligible costs relative to the same timescales at which SDN flow-level decision making operates, and fully absorbed within the TSM polling cycle of 500 *ms*, and Open vSwitch's flow setup budget of between 5 *ms* to 50 *ms*.

There are three expansion directions: hardware validation on the Raspberry Pi 4 and ESP32 to characterize TDCR-SDN on realistic IoT resource constraints; integration of certificateless signcryption-based trust attestation [19] that provides a proactive cryptographic trust signal over a static 3-second behavioral detection window, thereby shortening it; and an adaptive τ^* adjustment mechanism that closes the residual security gap during the detection interval without increasing false-positive suppression of legitimate principals.

References

[1] A. Rahdari, A. Jalili, M. Esnaashari, M. Gheisari, A. A. Vorobeva, Z. Fang, "Security and privacy challenges in SDN-enabled IoT systems", Computer Modeling in Engineering & Sciences, Vol.80, No.2, (2024), pp.2511–2533, available online: https://doi.org/10.32604/cmc.2024.052994

[2] Jimmington, Anjana, Sabu M. Thampi, Preetam Mukherjee, "Advancing Connectivity and Security in SDN-IoT Networks Through Policy Management", Proceedings of the conference "Securing the Connected World: Exploring Emerging Threats and Innovative Solutions", (2025), pp.223–261, available online: https://doi.org/10.1007/978-3-031-82826-3_7.

[3] D. Bringhenti, J. Yusupov, A. Zarca, Valenza F., Sisto R., Bernabe, J. B., & Skarmeta, A. "Automatic, verifiable and optimized policy-based security enforcement for SDN-aware IoT networks", Computer Networks, Vol.213, No.X, (2022), pp.109123, available online: https://doi.org/10.1016/j.comnet.2022.109123

[4] W. Bekri, R. Jmal, L. Chaari Fourati (2025), Distributed secured and trustworthy IoT framework based on SDN, AI, and blockchain. Cluster Computing 28, 1065, available online: https://doi.org/10.1007/s10586-025-05770-7

[5] A. D. Ferguson, A. Guha, C. Liang, R. Fonseca, S. Krishnamurthi, "Hierarchical policies for software defined networks", Proceedings of the 1st ACM SIGCOMM Workshop on Hot Topics in Software Defined Networks (HotSDN), (2012), pp.37–42, available online: https://doi.org/10.1145/2342441.2342450

[6] J. Reich, C. Monsanto, N. Foster, J. Rexford, D. Walker, "Modular SDN programming with Pyretic", USENIX Technical Report, Vol.38, No.5, (2013), pp.40–47.

[7] B. Lantz, B. Heller, N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks", Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets-IX), (2010), pp.1–6, available online: https://doi.org/10.1145/1868447.1868466

[8] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, & J. Turner, "OpenFlow: Enabling innovation in campus networks", ACM SIGCOMM Computer Communication Review, Vol.38, No.2, (2008), pp.69–74, available online: https://doi.org/10.1145/1355734.1355746

[9] K. K. Karmakar, V. Varadharajan, M. Hitchens, U. Tupakula, P. Sariputra, "A trust-aware OpenFlow switching framework for software defined networks (SDN)", Computer Networks, Vol.237, No.X, (2023), pp.110109, available online: https://doi.org/10.1016/j.comnet.2023.110109

[10] R. Aryan, A. Yazidi, A. Bouhoula, & P. E. Engelstad, (2025), A formal technique for automatic resolution of OpenFlow anomalies. International Journal of Information Security 24, 181, available online: https://doi.org/10.1007/s10207-025-01035-x

[11] M. Hussain, R. Amin, R. Gantassi, A. H. Alshehri, J. Frnda, & S. M. Raza, "Efficient handling of ACL policy change in SDN using reactive and proactive flow rule installation", Scientific Reports, Vol.14, No.1, (2024), available online: https://doi.org/10.1038/s41598-024-65721-x

[12] RYU Development Team, "RYU SDN framework, release 4.34", (2023), available online: https://ryu-sdn.org

[13] R. Aryan, A. Yazidi, Ø. Kure, P. E. Engelstad, "A parallel approach for detecting OpenFlow rule anomalies based on a general formalism", Concurrency and Computation: Practice and Experience, Vol.33, No.15, (2021), pp.e5907, available online: https://doi.org/10.1002/cpe.5907

[14] Liang, S., Su, J. (2023). Detection of SDN Flow Rule Conflicts Based on Knowledge Graph. In: Quan, W. (eds) Emerging Networking Architecture and Technologies. ICENAT 2022. Communications in Computer and Information Science, vol 1696. Springer, Singapore. https://doi.org/10.1007/978-981-19-9697-9_8

[15] Q. A. Arshad, W. Z. Khan, F. Azam, M. K. Khan, H. Yu, Y. B. Zikria (2023), Blockchain-based decentralized trust management in IoT: systems, requirements and challenges. Complex & Intelligent Systems 9, no (6) (2023), 6155–6176, available online: https://doi.org/10.1007/s40747-023-01058-8

[16] A. Irshad, G. A. Mallah, M. Bilal, S. A. Chaudhry, M. Shafiq and H. Song, "SUSIC: A Secure User Access Control Mechanism for SDN-Enabled IIoT and Cyber–Physical Systems," in *IEEE Internet of Things Journal*, vol. 10, no. 18, pp. 16504-16515, 15 Sept.15, 2023, doi: 10.1109/JIOT.2023.3268474

[17] S. Pisharody, J. Natarajan, A. Chowdhary, A. Alshalan and D. Huang, "Brew: A Security Policy Analysis Framework for Distributed SDN-Based Cloud Environments," in *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 6, pp. 1011-1025, 1 Nov.-Dec. 2019, doi: 10.1109/TDSC.2017.2726066.

[18] S. R. Mishra, B. Shanmugam, K. C. Yeo, S. Thennadil (2025), SDN-Enabled IoT Security Frameworks—A Review of Existing Challenges. *Technologies* 13(3), 121, available online: https://doi.org/10.3390/technologies13030121

[19] C. Zhou, Z. Zhao, W. Zhou, Y. Mei, "Certificateless key-insulated generalized signcryption scheme without bilinear pairings", Security and Communication Networks, Vol.2017, (2017), Article ID 8405879, available online: https://doi.org/10.1155/2017/8405879

[20] J. Xie, H. Tang, T. Huang, F. R. Yu, R. Xie, J. Liu, Y. Liu, "A survey of blockchain technology applied to smart cities: Research issues and challenges", in *IEEE Communications Surveys & Tutorials*, Vol.21, No.3, (2019), pp.2794–2830, available online: https://doi.org/10.1109/COMST.2019.2899617

[21] R. Soulé, S. Basu, P. J. Marandi, F. Pedone, R. Kleinberg, E. G. Sirer, "Merlin: A language for managing network resources", *in IEEE/ACM Transactions on Networking*, Vol.26, No.5, (2018), pp.2188–2201, available online: https://doi.org/10.1109/TNET.2018.2872900, last visit: 05.05.2026

[22] ONOS Project, Open Networking Foundation, (2022), available online: https://opennetworking.org/onos/, last visit: 05.05.2026.