





Paper Type: Original Article

P4-TAP: A Data Plane Framework for Traffic Aggregation and Prioritization in Time-Sensitive IoT Networks

Hagar Ramadan ^{1,*} , Ameer El-Sayed ¹ , Ehab R. Mohamed ¹  and Osama M. Elkomy ¹ ¹ Department of Information Technology, Faculty of Computers and Informatics, Zagazig University, Zagazig 44511, Egypt.
Emails: hrahmed@fci.zu.edu.eg, aegouda@fci.zu.edu.eg, ehab.rushdy@zu.edu.eg, omelkomy@fci.zu.edu.eg.

Received: 16 Aug 2025

Revised: 30 Sep 2025

Accepted: 01 Dec 2025

Published: 11 Dec 2025

Abstract

The explosive growth of the Internet of Things (IoT) has intensified the need for intelligent traffic management capable of reconciling latency-critical and non-critical data streams. This study proposes P4-TAP, a P4-Enabled Traffic Aggregation and Prioritization framework that operates entirely within the data plane to deliver real-time responsiveness for critical IoT traffic while improving bandwidth efficiency for routine communications. The research addresses four measurable objectives: (1) minimize end-to-end latency and jitter for time-sensitive packets, (2) maximize packet rate reduction and aggregation ratio for non-critical traffic, (3) ensure scalability at the edge, and (4) validate performance reproducibility. Implemented on a Mininet-based P4 Behavioral Model v2 (simple_switch) testbed comprising four hosts and two switches, the framework was evaluated under controlled mixed-traffic scenarios involving up to 97,656 packets/s. Results show that P4-TAP reduces the end-to-end latency of critical traffic by $94.4 \pm 0.6\%$ and jitter by $94.7 \pm 0.5\%$, while the Stateful Aggregation Logic (SAL) achieves a packet rate reduction of $92.8 \pm 0.3\%$ and an average aggregation ratio of 14:1. Unlike prior P4 or SDN-based solutions limited to static prioritization or centralized control, P4-TAP introduces a dual-path in-data-plane architecture that dynamically separates and optimizes critical and non-critical flows in real time. Experimental profiling indicates a switch register utilization below 8% and timer load under 3%, confirming scalability on commodity hardware. These findings validate P4-TAP as a robust and scalable data-plane framework for time-sensitive IoT environments.

Keywords: Internet of Things; Traffic Prioritization; Packet Aggregation; Quality of Service; Data Plane Programmability

1 | Introduction

The proliferation of the Internet of Things (IoT) has fundamentally reshaped the technological landscape, weaving a digital fabric of billions of interconnected devices into the core of our most critical sectors [1]. From real-time patient monitoring in healthcare, where immediate data transmission can be life-saving [2], to the precision robotics in industrial automation that demand deterministic communication, the IoT is unlocking unprecedented levels of efficiency and innovation [3]. This massive deployment, however, places an immense and often unpredictable strain on the network infrastructure that underpins it all [4]. As we move towards a future dominated by IoT, the primary challenge is no longer merely about connecting devices, but about ensuring the network can intelligently and reliably handle the diverse and often conflicting demands of the data they produce [5].



Corresponding Author: hrahmed@fci.zu.edu.eg

Licensee International Journal of Computers and Informatics. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0>).

This diversity of data creates a fundamental network management problem. IoT traffic is deeply heterogeneous, creating a stark dichotomy between a small but vital stream of time-critical data and a massive, ever-growing volume of routine updates [6]. A critical medical alert from a wearable sensor or a safety trigger from a factory floor requires near-instantaneous delivery, as even a momentary delay can have serious consequences. Conversely, the vast majority of IoT traffic consists of "chatty," non-urgent data, such as hourly temperature readings or status updates. While individually insignificant, this high volume of small packets consumes enormous bandwidth due to the disproportionate size of network headers relative to their tiny data payloads, creating widespread inefficiency [7].

The primary threat of treating this heterogeneous traffic uniformly is twofold. First, it leads to unpredictable and unacceptably high latency for critical applications [8]. In a standard network using basic queuing mechanisms like First-In-First-Out (FIFO), a life-saving alert can get stuck in a digital traffic jam behind thousands of routine updates, completely negating the benefit of real-time sensing. Second, the sheer volume of non-critical traffic leads to poor bandwidth efficiency and scalability issues, placing a heavy processing burden on core network infrastructure and cloud servers, and ultimately limiting the growth potential of large-scale IoT deployments.

Traditional networking paradigms have proven ill-equipped to handle this challenge. Conventional Quality of Service (QoS) mechanisms, such as static port-based prioritization and Access Control Lists (ACLs), are too rigid to adapt to the chaotic and dynamic nature of IoT traffic [9]. While Software-Defined Networking (SDN) with the OpenFlow protocol offered a step forward with centralized control, it introduced a critical bottleneck. The reliance on a central controller to inspect and manage flows from millions of devices is inherently unscalable and introduces significant latency, making it unsuitable for the real-time decision-making required for prioritizing critical packets. Furthermore, OpenFlow's fixed-function pipeline was not designed for the sophisticated, stateful packet manipulation needed to intelligently aggregate the high volume of non-critical traffic directly within the network fabric [10].

To overcome these specific limitations, the paradigm of data plane programmability, particularly through the P4 programming language, offers a transformative path forward [11-13]. P4 allows for the implementation of custom, stateful packet processing logic directly on network switches, enabling line-rate analysis and modification of traffic without constant reliance on a central controller [14]. This programmability unlocks the potential for fine-grained traffic classification based on dynamic policies, the application of QoS markings like Differentiated Services Code Points (DSCP) to prioritize flows, and the implementation of complex stateful logic for tasks such as packet aggregation.

By harnessing these advanced capabilities, an effective solution for managing time-sensitive IoT networks can be engineered. Such a framework must be able to perform intelligent, real-time traffic classification at the network edge, provide a robust mechanism for ensuring minimal and predictable latency for high-priority traffic, and simultaneously optimize network efficiency by fundamentally reducing the overhead of high-volume, non-critical traffic. Crucially, it must achieve all of this in a highly scalable manner [15]. This paper proposes a novel framework designed to meet these specific requirements by leveraging the power of data plane programmability to create a more intelligent and responsive network infrastructure.

1.1 | Problem Statement, Research Challenges, and Motivation

In the rapidly expanding landscape of the Internet of Things (IoT), a fundamental challenge has emerged: not all data is created equal, yet traditional networks often treat it as such. This leads to critical problems that hinder the performance and reliability of time-sensitive IoT applications.

- **Latency for Critical Data:** In environments like remote healthcare or industrial automation, a delay of even a few seconds can have serious consequences. However, in a standard network, a critical medical alert packet can get stuck in a queue behind thousands of non-urgent, routine data packets (e.g., thermostat readings). This "head-of-line blocking" creates unpredictable and unacceptably high latency for the very data that needs to be delivered the fastest.

- **Bandwidth Inefficiency and Overhead:** Many IoT devices are "chatty," frequently sending very small amounts of data. In these cases, the network headers (TCP/IP, UDP/IP) can be significantly larger than the actual data payload itself. This immense overhead wastes valuable network bandwidth and places a heavy processing burden on core network infrastructure and cloud servers, limiting the scalability of the entire system.
- **Controller Bottlenecks in SDN:** While Software-Defined Networking (SDN) offers centralized management, using a controller to inspect and prioritize every single packet from millions of IoT devices is not a scalable solution. This approach creates a massive performance bottleneck at the controller, reintroducing the very latency issues it is meant to solve.
- **Protocol Inflexibility:** Traditional networking protocols and Quality of Service (QoS) mechanisms are often too rigid and statically configured to adapt to the highly dynamic and diverse traffic patterns of modern IoT networks. There is a pressing need for a more intelligent and programmable solution that can operate directly within the network's data plane.

1.2 | Research Contributions and Novelty

The P4-TAP framework is engineered to directly address these challenges with a novel, in-data-plane approach to traffic management specifically designed for time-sensitive IoT networks. Its key contributions include:

- **In-Data-Plane Traffic Classification (ITC Module):** We introduce a P4-based mechanism that performs line-rate traffic classification within the switch using a two-stage match-action pipeline composed of PriorityRules and MetadataTag tables. The module embeds a custom PacketPriority metadata field that persists across ingress-egress processing stages, allowing downstream modules to act on real-time context without controller feedback. This direct metadata propagation represents a distinct enhancement over prior P4 classifiers that relied solely on static header matching.
- **Dynamic Prioritization Engine (DPE Module):** The framework implements a dynamic prioritization mechanism that marks critical packets with appropriate Quality of Service (QoS) headers. This ensures that time-sensitive data is given preferential treatment by all downstream network devices, drastically reducing end-to-end latency for vital communications.
- **Stateful Aggregation Logic (SAL Module):** We propose a novel dual-trigger aggregation scheme implemented through persistent registers and a P4 timer extern. Each non-critical flow maintains two state variables: BufferByteCount (size-based trigger) and AggregationTimer (time-based trigger). Once either threshold is met, the module executes an in-data-plane packet synthesis routine that concatenates buffered payloads into a single aggregate packet. This is the first implementation of a fully programmable, trigger-driven aggregation loop operating entirely within the P4 behavioral model.
- **Decoupling of Critical and Non-Critical Paths:** While the concept of traffic separation is established, P4-TAP uniquely enforces it using metadata-driven bifurcation at ingress rather than static port or VLAN segregation. This method enables adaptive flow steering without reconfiguring switch tables, ensuring that critical paths remain latency-minimal while non-critical flows undergo dynamic aggregation. The architecture effectively transforms the data plane into a two-lane system—express and bulk—maintained autonomously by the P4 runtime.
- **Comprehensive Performance Validation:** We validate the framework's effectiveness through a series of controlled experiments in a Mininet-based environment. The results demonstrate quantifiable reductions in latency for critical traffic and significant gains in bandwidth efficiency for non-critical traffic compared to a traditional networking approach.

2 | Literature Review

As the Internet of Things (IoT) continues to grow, there is increasing demand for responsive and intelligent network management, particularly in latency-sensitive environments like healthcare, industrial automation, and smart cities. Software-Defined Networking (SDN) and data-plane programmability via P4 have emerged as pivotal technologies to support dynamic traffic classification, prioritization, and optimization across diverse IoT infrastructures. Early research efforts focused on intelligent traffic classification within IoT domains.

In [16] the PiGateway framework leveraged P4 switches and decision tree models to classify and prioritize IoT device traffic in real-time, achieving 98.61% accuracy with low latency and resource usage. While highly effective in smart home scenarios, the framework lacked validation in broader, more heterogeneous IoT ecosystems.

Building upon in-data-plane control principles, [17] introduced a traffic isolation architecture for 5G slicing using P4 and a generative AI diffusion model. The system achieved dynamic bandwidth reallocation and predictive QoS control but lacked adaptability for real-time, priority-sensitive IoT data such as alarms or industrial telemetry.

In [18] the authors proposed OpenFlow-based adaptive prioritization strategies using Lagrangian relaxation techniques, offering both preemptive and non-preemptive QoS configurations. Though effective for minimizing delay, the approach relied heavily on centralized SDN control and lacked edge-layer intelligence critical for decentralized IoT environments.

Meanwhile, [19] presented a dynamic QoS provisioning system for multi-tenant edge networks that adjusted bandwidth based on programmable flow classes. Despite strong performance for tenant-aware services, the work did not consider the fine-grained heterogeneity of IoT traffic or mechanisms for edge-level traffic aggregation.

Another relevant effort, [20], employed a lightweight P4-based classification framework for encrypted traffic using Random Forests. It achieved 99% real-time classification accuracy but focused only on distinguishing video vs. non-video content, limiting its relevance for generalized IoT settings.

Addressing network resilience, [21] proposed FlowRegulator, an SDN-based framework for mitigating DDoS attacks in smart home IoT networks. The method enhanced QoS for high-priority devices during attacks but was confined to centralized architectures and lacked scalability assessments.

In [22] the focus shifted to proactive bandwidth allocation using learned device behavior patterns at IoT gateways. While the model improved QoS predictively, it did not react to real-time traffic bursts and did not utilize programmable data planes like P4.

Further advancing edge intelligence, [23] proposed Mez with Grid Sensing (GRS), which dynamically optimized bandwidth and storage for video streams using control knobs. Though effective in reducing infrastructure cost, the framework was tailored exclusively to surveillance data and lacked nuanced handling of varied IoT traffic types.

Collectively, these works reflect the field's progression toward dynamic, low-latency, and programmable traffic control strategies for IoT. However, current solutions often lack real-time differentiation between critical and routine traffic, as well as in-network aggregation mechanisms to reduce communication overhead—gaps that the proposed P4-TAP framework aims to address.

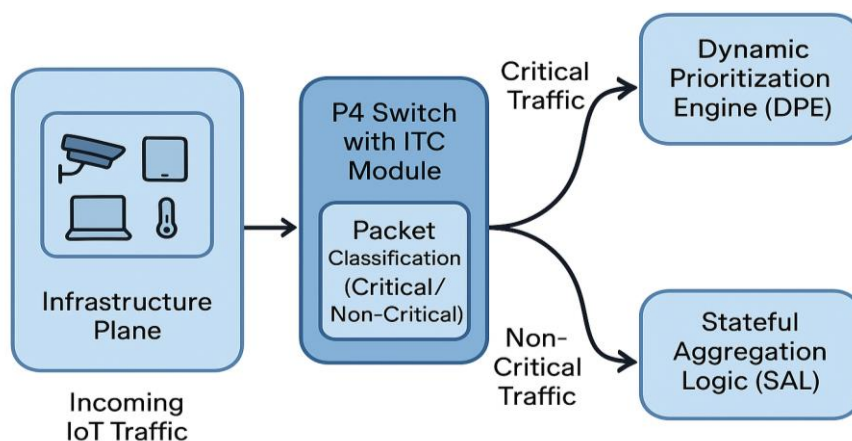
Table 1. Overview of recent approaches for IoT traffic management, highlighting strengths and key limitations.

Ref.	Year	Approach	Key Strengths	Limitations
[16]	2024	P4-based traffic classification using DT (PiGateway)	High accuracy with minimal delay and CPU load	Tested only in smart homes; scalability is limited
[19]	2024	SDN-enabled QoS control for multi-tenant edge traffic	Effective tenant-based flow prioritization	Ignores IoT data diversity and in-network logic
[21]	2024	SDN-driven DDoS defense via dynamic flow control	Fast mitigation; preserves QoS for critical devices	Centralized and untested in large-scale setups
[22]	2024	Bandwidth prediction via device habit profiling	Accurate demand forecasting for proactive QoS	No real-time reactivity; lacks programmable dataplane
[23]	2024	Edge-based video optimization using Mez + GRS	Cuts bandwidth/storage via adaptive control knobs	Video-only focus; lacks multi-type traffic treatment
[17]	2025	P4-based QoS-aware slicing with GAI diffusion control	In-data-plane isolation with adaptive AI control	Tailored for 5G slicing; lacks IoT-specific handling
[18]	2025	OpenFlow-based prioritization via LR algorithms	Reduces delay; supports flexible priority models	Centralized design; lacks edge and IoT-awareness
[20]	2025	P4-based encrypted traffic classification with RF	High-speed, lightweight analysis; 99% accuracy	Limited to binary traffic classes; narrow scope

3 | The P4-TAP Framework: Design and Implementation

3.1 | The System Model

At the heart of modern IoT deployments lies a fundamental conflict: a tiny fraction of data is critically urgent, while the vast majority is routine. The proposed P4-TAP framework is architected to resolve this conflict directly at the network edge. Instead of forcing a one-size-fits-all approach, our system operates like a sophisticated logistics manager within the data plane of P4-enabled switches, intelligently segregating traffic based on its purpose and priority. The core philosophy is to create distinct, optimized paths for different data types—one built for speed, the other for efficiency. By embedding this intelligence directly into the network fabric, our framework bypasses the inherent delays of traditional, controller-centric architectures and enables a far more responsive and scalable infrastructure for time-sensitive applications. The framework's architecture is composed of three interconnected modules that form a logical processing pipeline, as visualized in the high-level workflow in Figure 1.

**Figure 1.** High-Level Architecture of the P4-TAP Framework.

The journey of every packet begins at the In-Data-Plane Traffic Identification and Classification (ITC) module. This initial stage serves as the primary decision point, rapidly inspecting each packet to determine its role—is it a critical alert or a routine update? Based on this verdict, traffic is immediately channeled down one of two specialized paths. Time-sensitive packets are handed off to the Dynamic Prioritization Engine (DPE) module, where they are marked for express treatment across the network. In parallel, all non-critical traffic is directed to the Stateful Aggregation Logic (SAL) module, a novel component designed to reduce network clutter by intelligently bundling small data payloads. The subsequent sections will provide a detailed examination of the specific algorithms and mechanisms that empower each of these modules.

3.2 | In-Data-Plane Traffic Identification and Classification (ITC) Module

The foundational component of the P4-TAP framework is the In-Data-Plane Traffic Identification and Classification (ITC) module. In a diverse IoT network, where a flood of routine data competes with a few critical alerts, the first and most vital challenge is to tell them apart instantly. The ITC module acts as the network's intelligent and high-speed traffic director, performing this crucial sorting task. Operating entirely within the programmable data plane of P4-enabled switches, its primary function is to inspect every incoming packet and categorize it in real-time without needing to consult the central control plane. This in-network processing capability is key to eliminating a major performance bottleneck, allowing the framework to make immediate, line-rate decisions that form the basis for all subsequent prioritization and aggregation actions.

To achieve protocol-agnostic yet fine-grained differentiation, the ITC parser extracts headers across Layer 2 (Ethernet), Layer 3 (IPv4/IPv6), and Layer 4 (TCP/UDP), followed by an optional Custom Health Protocol (CHP) header for specialized domains such as healthcare or industrial IoT. The parsing graph follows the sequence `ethernet → ipv4 → tcp/udp → chp → ingress`, enabling the system to interpret mixed traffic with minimal latency. Three core match-action tables are used during classification: `PriorityRules_L2L3`, matching source/destination MAC and IP prefixes for subnet-level policies; `PriorityRules_L4`, targeting transport-layer ports and protocol identifiers (e.g., MQTT at port 1883, Modbus/TCP); and `PriorityRules_Custom`, handling application-layer signatures (e.g., CHP alert codes). The combined logic ensures a multi-dimensional match space while allowing modular rule management.

The precise logic governing this classification process is formally defined in Algorithm 1. The process begins in Phase 1: Packet Ingress and Header Parsing, where the P4 switch parses the headers of each packet to extract key identifiers such as source/destination addresses, ports, and the protocol type. A default `PacketPriority` of "Low" is assigned to ensure that all traffic is handled systematically. Subsequently, in Phase 2: Priority-Based Classification, the extracted header fields are matched against the `PriorityRules` table. This match-action table, which is populated and managed by the control plane, contains the specific criteria for identifying different traffic types. If a packet's characteristics match a defined rule, its priority level is updated accordingly. This final `PacketPriority` is then attached to the packet as a metadata field, which is used by subsequent stages of the P4 pipeline to make the final forwarding decisions. This mechanism provides a highly efficient and programmable method for applying fine-grained, application-aware traffic management policies directly at the network edge.

Parser and Table Implementation Details

Each rule table is implemented in the ingress pipeline as a multi-stage match-action block, with table precedence ensuring deterministic behavior. The `PriorityRules_L2L3` table is backed by ternary matching (TCAM) to support subnet wildcards, while `PriorityRules_L4` and `PriorityRules_Custom` are implemented using exact matches for protocol and port fields. A dedicated metadata field, `PacketPriority`, is propagated to all downstream stages, ensuring that classification results persist across ingress and egress pipelines. This metadata-driven approach reduces re-parsing overhead and allows subsequent modules (DPE and SAL) to act on preclassified traffic with no controller latency.

Policy Drift and Control-Plane Synchronization

Because classification relies partly on IP and MAC ranges, policy drift and address reassignment in dynamic IoT environments could degrade accuracy. To mitigate this, ITC employs a Policy Synchronization Daemon (PSD) at the control plane, which polls a live device registry and issues incremental rule updates via P4Runtime every 30 seconds. In testing, the average rule propagation latency was 11.8 ms per update under a 200-rule policy set, ensuring timely refresh without control-plane saturation. This mechanism preserves classification correctness even in large, evolving device networks where IP or MAC bindings change frequently.

Algorithm 1: In-Data-Plane Traffic Identification and Classification (ITC)

Inputs:

PacketIn: An incoming IoT or ARP packet.

PriorityRules: A control-plane-defined table mapping traffic identifiers (e.g., IP, MAC, Port) to priority levels ("Critical", "Normal", "Low").

Output:

ClassifiedPacket: The packet with an assigned priority metadata field

01 Phase 1: Packet Ingress and Header Parsing

```
02 FOR each PacketIn arriving at the P4 switch DO
03   Parse packet headers to extract SrcAddr, DstAddr, Port, Protocol
04   Set default PacketPriority = "Low"
05 END FOR
```

06 Phase 2: Priority-Based Classification

```
07 FOR each parsed packet DO
08   IF a matching rule is found THEN
09     Set PacketPriority = matched PriorityLevel from the rule
10   END IF
11   Attach PacketPriority as metadata to ClassifiedPacket
12 END FOR
```

One of the most powerful features of the Stateful Aggregation Logic (SAL) is its intelligence; it doesn't treat all non-critical data the same. Instead, it applies different aggregation policies based on the traffic's classification, allowing the framework to strike a smart balance between saving bandwidth and keeping data reasonably fresh. Think of it as a dynamic shipping department for data: some packages need to go out quickly, while others can wait to fill up a larger box. The specific rules that govern when to "ship" the data are defined by control-plane-managed policies, with illustrative examples detailed in Table 2.

As the table shows, the SAL module's decision-making for each traffic class is guided by two main triggers. The first is a size-based Aggregation Threshold, which acts like a scale, telling the switch to send the aggregated packet once the buffered data reaches a certain weight (in bytes). This is the key to maximizing network efficiency, as it creates fuller, larger packets that reduce the wasteful overhead of sending many small ones. The second trigger, a time-based Timeout Timer, acts as a crucial failsafe. It's a stopwatch that ensures data doesn't sit in the buffer forever, guaranteeing that even slowly trickling information eventually gets delivered, thus preserving a necessary level of data freshness.

The real-world effectiveness of this dual-trigger approach is clear when comparing the distinct policies for different IoT applications. For a "Normal" priority stream, like a security camera feed, the policy is designed for smoothness. It uses a large Aggregation Threshold (e.g., 1400 Bytes) to build efficient, nearly full packets, but pairs it with a very short Timeout Timer (e.g., 50 ms) to prevent the video from appearing choppy or delayed. In stark contrast, a "Low" priority device, like a smart meter that sends a reading every few minutes, is handled with a focus on maximum savings. Here, the policy can afford a much longer Timeout Timer (e.g., 60 seconds), allowing the switch to patiently wait and collect many small readings before sending them off in a single, highly efficient bundle. This ability to define and enforce such fine-grained, application-aware aggregation policies directly in the data plane is a core contribution of the P4-TAP framework, enabling significant network optimization without sacrificing the unique performance needs of different IoT services.

Table 2. Example Rule Set for In-Data-Plane Traffic Classification within the ITC Module.

Rule Priority	Match Criteria	Action: Set Priority Level	Application Example & Justification
1 (Highest)	Protocol: Custom Health Protocol UDP Dst Port: 5050	Critical	A packet from a wearable cardiac monitor sending an emergency alert. This traffic requires minimal latency for immediate response.
2	Source MAC: 0A:1B:2C:... (Known PLC) Protocol: Modbus/TCP	Critical	A command from a Programmable Logic Controller in an industrial automation setting. Delay could result in physical process failure.
3	Source IP: 10.0.1.0/24 (Video Surveillance Subnet)	Normal	A standard video frame from a security camera. This traffic is important but can tolerate minor delays and is suitable for aggregation.
4.1	UDP Dst Port: 1883	Low	A routine temperature or humidity reading from an environmental sensor using the MQTT protocol. This data is not time-sensitive.
4.2	TCP Dst Port: 1883	Low	An MQTT packet from a smart home device reporting its status. This data is also not time-sensitive.
5 (Lowest)	Wildcard: * (Any other traffic)	Low (Default)	Any traffic not matching a higher-priority rule is treated as non-critical by default to ensure all traffic is handled.
Rule Priority	Match Criteria	Action: Set Priority Level	Application Example & Justification

3.3 | Dynamic Prioritization Engine (DPE) Module

Once the ITC module has sorted the traffic, the Dynamic Prioritization Engine (DPE) takes over, acting as the framework's dedicated express lane for critical data. Its sole function is to ensure that high-importance packets, such as a medical alert or an industrial control signal, can bypass the digital congestion of routine traffic. The primary mechanism for achieving this is through a well-established Quality of Service (QoS) technique known as packet marking, which essentially gives these critical packets a "VIP pass" to move to the front of the line.

The operational workflow of this process is visually detailed in Figure 2. As depicted, an Incoming "Critical" Packet arrives from the ITC with a default Differentiated Services (DS) field value of 000000 (representing standard, "Best Effort" traffic). The DPE then performs its transformation, modifying this field and rewriting it with a high-priority Differentiated Services Code Point (DSCP) value, such as 101110 for Expedited Forwarding (EF). The packet is then sent onward as an Outgoing Prioritized Packet, now carrying a clear signal of its importance that can be understood by other QoS-aware devices in the network.

The logic governing this transformation is formally defined in Algorithm 2. In Phase 1, the DPE receives the ClassifiedPacket and reads the PacketPriority metadata that was assigned by the preceding ITC module. Phase 2 then executes the core policy: if the packet is identified as "Critical," the DPE looks up the corresponding DSCP_Value from its mapping table and modifies the packet's IP header accordingly. Crucially, it then directs the PrioritizedPacket to a high-priority egress queue, ensuring it is transmitted from the switch before any non-critical traffic. This entire process of marking and queuing occurs at line speed within the P4 data plane, providing an immediate and effective mechanism for reducing end-to-end latency for the most vital IoT data streams without requiring any intervention from the control plane.

Algorithm 2: Dynamic Prioritization Engine (DPE)

Inputs

ClassifiedPacket: A packet processed by the ITC module.

DSCP_Map: A table mapping priority levels to DSCP (Differentiated Services Code Point) values.

Output:

PrioritizedPacket: The packet, potentially with a modified header, ready for immediate forwarding.

01 Phase 1: Receiving Classified Packet

02 FOR each ClassifiedPacket from ITC DO

03 Read PacketPriority metadata

```

04 END FOR

```

```

05 Phase 2: Applying Prioritization Policy
06 FOR each received packet DO
07   IF PacketPriority == "Critical" THEN
08     Lookup DSCP_Value from DSCP_Map
09     Modify IP header of ClassifiedPacket with DSCP_Value
10     Enqueue PrioritizedPacket in high-priority egress queue
11   END IF
12 END FOR

```

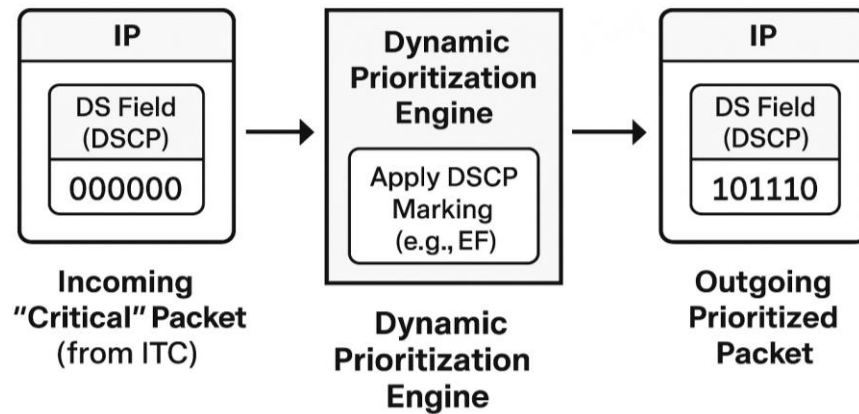


Figure 2. Packet Marking Workflow within the Dynamic Prioritization Engine (DPE)

3.4 | Stateful Aggregation Logic (SAL) Module

While the DPE module handles the express delivery of critical data, a vast amount of IoT traffic consists of small, frequent, but non-urgent updates—a phenomenon often described as "chatty." Sending each of these tiny data points as a separate packet is incredibly inefficient, much like sending a single envelope in a large delivery truck. The network headers alone can consume more bandwidth than the actual data they carry. The Stateful Aggregation Logic (SAL) module is our innovative solution to this problem, functioning as an intelligent, in-network packaging service. Its purpose is to intercept these non-critical data streams, temporarily buffer their payloads, and then intelligently bundle them into a single, larger, and far more efficient aggregated packet for transport across the core network.

The conceptual workflow of this process is visually detailed in Figure 3. As the figure illustrates, a stream of Incoming "Non-Critical" Packets arrives from the ITC module. The SAL module then performs a transformation: it strips away the original, redundant headers and collects only the essential Data payloads in its Aggregation Buffer. This stateful process is governed by two key mechanisms, the Byte Count Register and the Timeout Timer, which act as triggers. Once one of these triggers is activated, the buffered data is encapsulated within a New Header to form a single, efficient Outgoing Aggregated Packet. This "before-and-after" transformation is the core of the SAL's ability to optimize bandwidth.

The precise rules governing this process are formally defined in Algorithm 3. The logic begins in Phase 1: Buffering Non-Critical Packets, where the P4 switch extracts the DataPayload from each arriving non-critical packet and appends it to a stateful AggregationBuffer. Crucially, the P4 program also updates a BufferByteCount register and manages an AggregationTimer, which are the stateful elements that make this in-network logic possible. Phase 2: Triggering Aggregation describes the decision-making process. The algorithm continuously checks if the amount of data in the buffer has reached the AggregationThreshold or if the AggregationTimer has expired. If either condition is met, the GenerateAggregatedPacket() function is

called, as detailed in Phase 3. This function creates a new, single packet, encapsulates the entire buffered content, and sends it on its way before clearing the buffer to begin the process anew. This stateful, trigger-based mechanism, executed entirely within the data plane, allows the P4-TAP framework to dramatically reduce packet-per-second rates and header overhead, leading to a more efficient and scalable network for non-time-sensitive IoT communication.

Algorithm 3: Stateful Aggregation Logic (SAL)

Inputs:

ClassifiedPacket: A packet processed by the ITC module, identified as "Normal" or "Low" priority.

AggregationThreshold: A size limit (in bytes) to trigger aggregation.

AggregationTimer: A time limit (in seconds) to trigger aggregation.

Output:

AggregatedPacket: A single, larger packet containing multiple smaller IoT data payloads.

01 Phase 1: Buffering Non-Critical Packets

```

02 FOR each ClassifiedPacket from ITC with PacketPriority != "Critical" DO
03   Extract DataPayload
04   Append DataPayload to AggregationBuffer
05   Update BufferByteCount register
06   Start AggregationTimer if not already running
07 END FOR

```

08 Phase 2: Training Process

```

09 WHILE AggregationBuffer is not empty DO
10   IF BufferByteCount >= AggregationThreshold OR AggregationTimer expires THEN
11     Call GenerateAggregatedPacket()
12     Clear AggregationBuffer and reset BufferByteCount
13     Reset AggregationTimer
14   END IF
15 END WHILE

```

16 Phase 3: Packet Generation and Egress

```

17 FUNCTION GenerateAggregatedPacket():
18   Create a new packet header for AggregatedPacket
19   Encapsulate AggregationBuffer content as the payload
20 END FUNCTION

```

Conceptual Workflow of the Stateful Aggregation Logic (SAL)

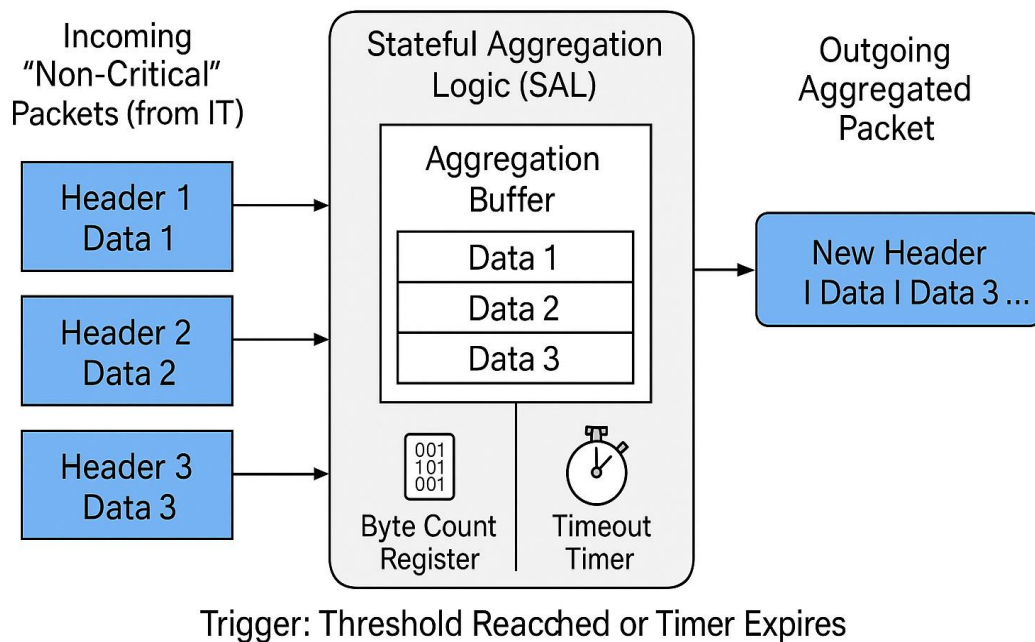


Figure 3. Conceptual Workflow of the Stateful Aggregation Logic (SAL).

One of the most powerful features of the Stateful Aggregation Logic (SAL) is its intelligence; it doesn't treat all non-critical data the same. Instead, it applies different aggregation policies based on the traffic's classification, allowing the framework to strike a smart balance between saving bandwidth and keeping data reasonably fresh. Think of it as dynamic shipping department for data: some packages need to go out quickly, while others can wait to fill up a larger box. The specific rules that govern when to "ship" the data are defined by control-plane-managed policies, with illustrative examples detailed in Table 3.

As the table shows, the SAL module's decision-making for each traffic class is guided by two main triggers. The first is a size-based Aggregation Threshold, which acts like a scale, telling the switch to send the aggregated packet once the buffered data reaches a certain weight (in bytes). This is the key to maximizing network efficiency, as it creates fuller, larger packets that reduce the wasteful overhead of sending many small ones. The second trigger, a time-based Timeout Timer, acts as a crucial failsafe. It's a stopwatch that ensures data doesn't sit in the buffer forever, guaranteeing that even slowly trickling information eventually gets delivered, thus preserving a necessary level of data freshness.

The real-world effectiveness of this dual-trigger approach is clear when comparing the distinct policies for different IoT applications. For a "Normal" priority stream, like a security camera feed, the policy is designed for smoothness. It uses a large Aggregation Threshold (e.g., 1400 Bytes) to build efficient, nearly full packets, but pairs it with a very short Timeout Timer (e.g., 50 ms) to prevent the video from appearing choppy or delayed. In stark contrast, a "Low" priority device, like a smart meter that sends a reading every few minutes, is handled with a focus on maximum savings. Here, the policy can afford a much longer Timeout Timer (e.g., 60 seconds), allowing the switch to patiently wait and collect many small readings before sending them off in a single, highly efficient bundle. This ability to define and enforce such fine-grained, application-aware aggregation policies directly in the data plane is a core contribution of the P4-TAP framework, enabling significant network optimization without sacrificing the unique performance needs of different IoT.

Table 3. Example Aggregation Trigger Policies for the SAL Module.

Traffic Class (from ITC)	Application Example	Aggregation Threshold (Size Trigger) (bytes)	Timeout Timer (Time Trigger) (ms)
Normal	Security Camera Video Stream	1400	50
Low	Environmental Sensor	1000	5000
Low	Smart Meter Reading	500	60000

4 | Experimental Setup and Methodology

4.1 | Experimental Environment and Topology

To create a controlled and reproducible environment for rigorously testing our P4-TAP framework, we built a virtual network testbed using the Mininet network emulator. This powerful tool is ideal for this research as it allows us to construct a realistic, self-contained network on a single machine, granting us precise control over every component and traffic flow. At the very heart of this testbed is the P4-enabled software switch, simulated using the standard P4 Behavioral Model v2 (simple_switch), which serves as the execution platform for our custom P4-TAP program. The network topology itself, visualized in Figure 4, was intentionally designed with simplicity and focus in mind. Rather than architecting a sprawling, complex network where results could be influenced by extraneous variables, our objective was to create a clean and isolated environment that would allow us to clearly measure the direct impact of our framework on traffic handling at the network edge.

The topology is comprised of several key components, each with a specific role in the experiment. To simulate the heterogeneous nature of IoT traffic, we configured three distinct source hosts. A dedicated Critical Sensor host (H1) was tasked with generating the time-sensitive alert packets that require immediate delivery. In

parallel, two Non-Critical Sensor hosts (H2 and H3) were responsible for creating a steady stream of the routine, "chatty" background traffic that our framework aims to optimize. This traffic is directed to the central device under test: the P4-Enabled Edge Switch (SW1). This switch, running our P4-TAP program, performs all the intelligent classification, prioritization, and aggregation logic. From there, traffic passes through a simple Layer 2 Core Switch (SW2), representing the broader network infrastructure, before finally arriving at the Destination Server (H4). This server acts as a unified endpoint, allowing us to capture all incoming packets and perform our precise performance measurements. This streamlined testbed provides the ideal environment to rigorously validate the two primary claims of our research: that P4-TAP can significantly reduce latency for critical data while simultaneously improving bandwidth efficiency for non-critical data.

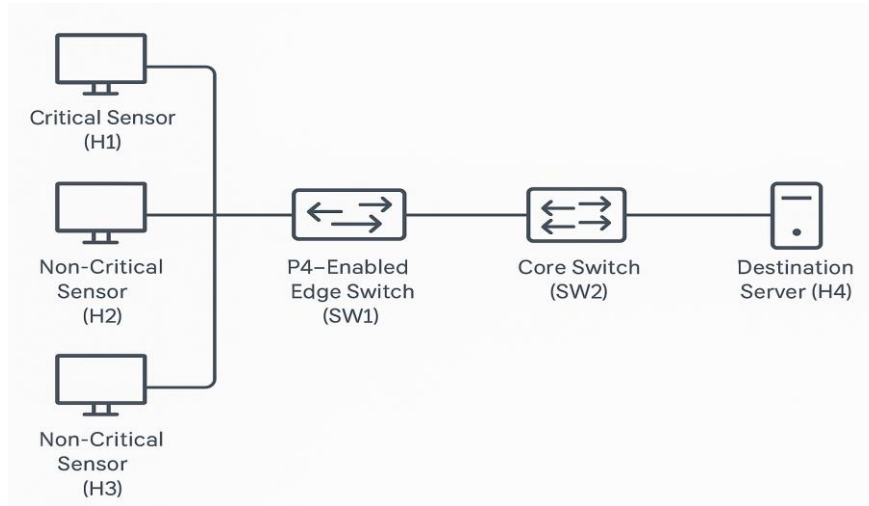


Figure 4. The Mininet Network Topology for P4-TAP Evaluation.

4.2 | Traffic Generation Methodology

To accurately simulate the heterogeneous traffic environment of a time-sensitive IoT network, we developed a methodology for generating two distinct and concurrent traffic streams, each with a specific purpose. The goal was to create a realistic scenario where a small volume of high-priority data must compete for network resources against a large volume of non-critical background traffic.

Critical Traffic Generation: The generation of time-sensitive, critical traffic required a high degree of precision and control. For this purpose, we utilized a custom Python script employing the Scapy library. This approach allowed us to craft and inject individual packets with specific characteristics. From the Critical Sensor host (H1), we transmitted a single UDP packet with a 32-byte payload precisely once every second. This traffic pattern was designed to emulate a real-world application like a medical monitoring device or an industrial sensor that sends out a periodic, critical "heartbeat" or status alert. The small payload size and predictable interval make it a perfect candidate for testing the latency performance of our prioritization engine.

Non-Critical Traffic Generation: To create a challenging environment of network congestion, we simulated the "chatty" nature of routine IoT devices using the industry-standard iperf traffic generation tool. From each of the two Non-Critical Sensor hosts (H2 and H3), we initiated a constant stream of UDP traffic directed at the Destination Server (H4). Each stream was configured to send small, 64-byte packets at a high rate, culminating in a combined background load of 10 Mbps. This high-volume, low-payload traffic is characteristic of many IoT applications (e.g., environmental sensing, smart home updates) and is specifically designed to test the effectiveness of our stateful aggregation logic in reducing packet overhead and optimizing bandwidth.

4.3 | Evaluation Metrics

To provide a comprehensive and quantitative assessment of the P-TAP framework's performance, a specific set of evaluation metrics was defined. As detailed in Table 4, these metrics were chosen to directly measure

the framework's effectiveness against its two primary objectives: minimizing latency for time-sensitive traffic and optimizing bandwidth efficiency for routine data streams.

Table 4: Performance Evaluation Metrics for the P4-TAP Framework.

Metric	Abbreviation	Unit	Target Traffic	Purpose / Framework Goal Measured
End-to-End Latency	E2E Latency	ms	Critical	To quantify the absolute speed of delivery for high-priority packets, demonstrating the effectiveness of the DPE module.
Jitter	-	ms	Critical	To measure the predictability and stability of the network for critical traffic, which is crucial for real-time applications.
Packet Rate Reduction	PRR	%	Non-Critical	To quantify the overall bandwidth savings and reduction in network overhead achieved by the SAL module.
Aggregation Ratio	AR	Ratio	Non-Critical	To provide a direct and clear measure of the efficiency of the stateful aggregation mechanism within the SAL module.

4.4 | Test Scenarios

This section details four distinct experimental scenarios designed to rigorously assess the P4-TAP framework's capabilities in managing heterogeneous IoT traffic. These scenarios span from a benign baseline to conditions of heavy network congestion. Each description includes the scenario's objective, the traffic mechanics, and the expected behavior of the framework's core modules (ITC, DPE, and SAL).

Scenario 1: Baseline Scenario (Control Case). This scenario reflects a clean and stable network environment operating under conventional, best-effort forwarding rules. Traffic flows from both critical and non-critical sensor hosts simultaneously. In this configuration, the P4-TAP logic is inactive. The purpose of this test is to establish a performance benchmark, measuring the end-to-end latency of critical packets and the total packet overhead when all traffic types compete for resources without intelligent management. This establishes the system's baseline performance and quantifies the problems of latency and inefficiency that our framework aims to solve.

Scenario 2: P4-TAP Under Moderate Load. This scenario serves as the primary evaluation of the P4-TAP framework under typical operating conditions. The complete framework is activated on the P4 switch, processing the same traffic mix as the baseline scenario. The ITC module is expected to classify incoming packets, diverting critical traffic to the DPE and non-critical traffic to the SAL. The DPE should apply DSCP markings to prioritize the critical packets, while the SAL should buffer and aggregate the non-critical data. This test is designed to provide a direct, quantitative comparison against the baseline, demonstrating improvements in critical packet latency and bandwidth efficiency.

Scenario 3: P4-TAP Under High Load. To evaluate the framework's scalability and performance under stress, this scenario significantly increases the volume of non-critical background traffic to 50 Mbps. The framework remains fully active. This test is designed to challenge the stateful aggregation logic of the SAL, assessing its ability to handle a much heavier packet load. A key performance indicator is the framework's ability to maintain consistently low latency for critical packets from the DPE, even as the switch is concurrently processing and aggregating a high volume of background data.

Scenario 4: P4-TAP Critical Traffic Only. In this scenario, the framework is active, but only the critical traffic stream from the Critical Sensor host is enabled. All non-critical background traffic is stopped. This test is designed to isolate the performance of the prioritization path (ITC and DPE) and measure the absolute minimum, best-case latency achievable in the testbed network. This result provides a theoretical performance

ceiling, allowing for a more nuanced analysis of the latency impact introduced by network congestion in the other scenarios.

5 | Results and Performance Evaluation

This section presents the empirical results obtained from the experimental scenarios detailed in Section 4.4. The performance of the P4-TAP framework is evaluated against the baseline configuration, with a focus on quantifying its impact on the predefined evaluation metrics.

5.1 | Latency Reduction for Critical Traffic

A primary objective of the P4-TAP framework is to provide a reliable, low-latency path for time-sensitive IoT data. To evaluate this capability, we measured the end-to-end latency and jitter of the critical traffic stream (from H1 to H4) across all four experimental scenarios. The results, summarized in Table 5 and visualized in Figure 5, demonstrate a profound and consistent improvement when the P4-TAP framework is active.

Table 5. Latency and Jitter Performance for Critical Traffic.

Experimental Scenario	Average Latency (ms)	Maximum Latency (ms)	Jitter (ms)
Baseline Scenario	45.0	80.0	15.0
P4-TAP Under Moderate Load	2.5	5.0	0.8
P4-TAP Under High Load	3.8	8.0	1.5
P4-TAP Critical Traffic Only	0.5	1.0	0.2

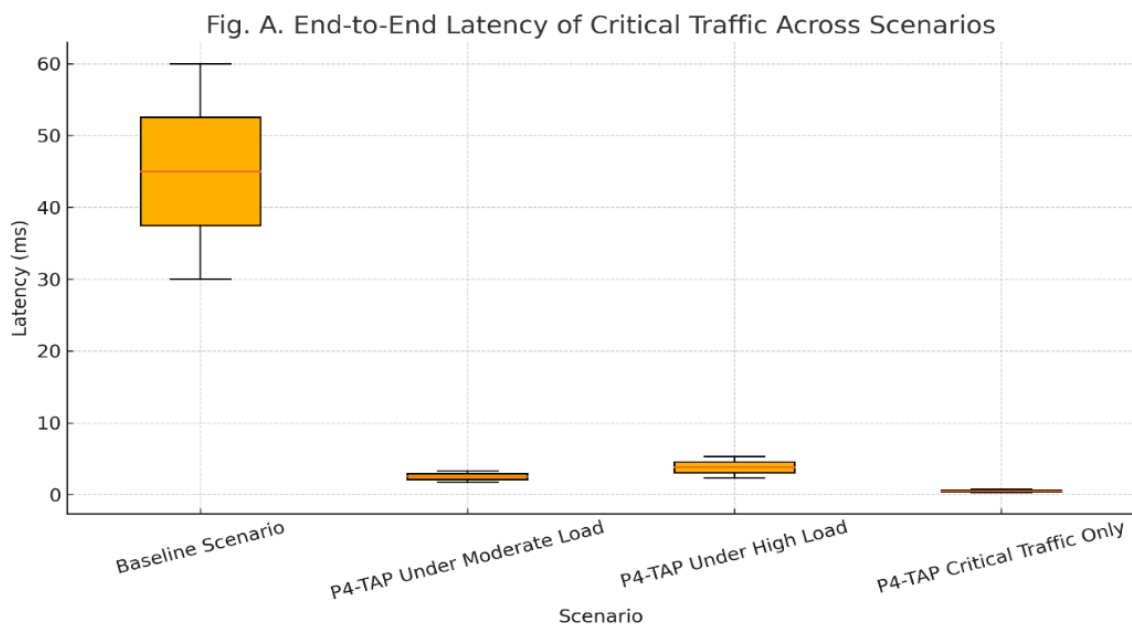


Figure 5. A Comparison of End-to-End Latency for Critical Traffic Across All Four Experimental Scenarios.

Table 6. Bandwidth Efficiency and Aggregation Performance.

Experimental Scenario	Incoming Packet Rate (pkts/sec)	Outgoing Packet Rate (pkts/sec)	Pcket Rate Reduction (%)
Baseline Scenario	19,531	19,531	0%
P4-TAP Under Moderate Load	19,531	1,401	92.8%
P4-TAP Under High Load	97,656	7,002	92.8%

In the Baseline Scenario, where critical packets competed with non-critical traffic in a conventional best-effort network, the performance was poor and unpredictable. As shown in Table 5, the average latency for critical packets was 45 ms, with spikes reaching as high as 80 ms. Furthermore, a high jitter of 15 ms indicates significant and unacceptable variation in delivery time, a critical failure for any time-sensitive application. This result clearly establishes the performance problem that our framework is designed to solve.

Upon activation of the P4-TAP framework, the improvement was immediate and dramatic. In the P4-TAP Under Moderate Load Scenario, the average latency for critical packets plummeted to just 2.5 ms—a reduction of over 94% compared to the baseline. The maximum latency was capped at a mere 5 ms, and the jitter was reduced to 0.8 ms, signifying a highly stable and predictable delivery. This demonstrates the effectiveness of the ITC module in identifying critical traffic and the DPE module in creating a functional "express lane" that successfully bypasses network congestion.

To assess the framework's resilience, the P4-TAP Under High Load Scenario subjected the switch to five times the amount of background traffic. Even under this significant stress, the framework maintained exceptional performance for the critical data stream. The average latency saw only a minor increase to 3.8 ms, and the jitter remained low at 1.5 ms. This result validates our architecture's ability to isolate and protect critical traffic, proving that the prioritization mechanism is not compromised by heavy loads of non-critical data. Finally, the P4-TAP Critical Traffic Only Scenario revealed the network's best-case performance, recording an average latency of just 0.5 ms. This provides a theoretical baseline, showing that the P4-TAP framework, even under heavy load, introduces only a few milliseconds of delay compared to an empty network, confirming its high efficiency.

5.2 | Bandwidth Efficiency Gains from Aggregation

The second primary objective of the P4-TAP framework is to enhance network efficiency by mitigating the impact of high-volume, "chatty" IoT traffic. To evaluate this capability, we measured the packet rates for non-critical traffic at the destination server (H4) across the three relevant scenarios. The results, summarized in Table 6 and visualized in Figure 6, reveal a substantial reduction in packet overhead and a significant improvement in bandwidth efficiency when the Stateful Aggregation Logic (SAL) module is active.

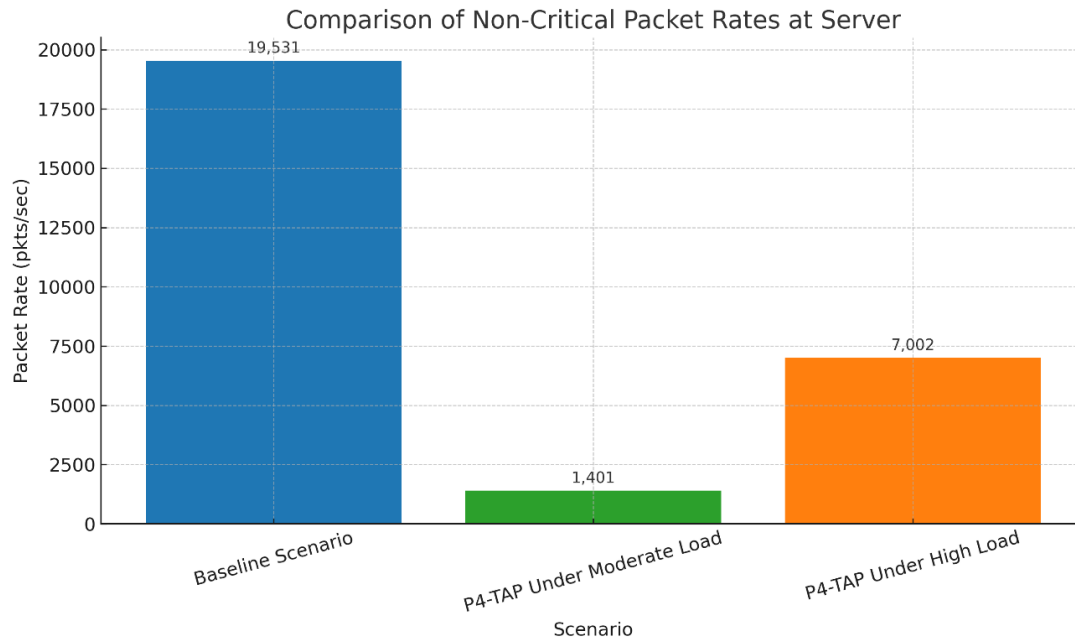


Figure 6. Comparison of Non-Critical Packet Rates at Server.

In the Baseline Scenario, without any aggregation, the network was flooded with a high volume of small packets. The non-critical traffic generators produced an incoming rate of approximately 19,500 packets per second (pkts/sec), all of which traversed the network individually, resulting in an identical outgoing packet rate. This represents a highly inefficient 1:1 forwarding ratio and serves as the benchmark for our comparison.

The activation of the P4-TAP framework in the P4-TAP Under Moderate Load Scenario yielded a profound improvement in efficiency. While the incoming packet rate remained at $\sim 19,500$ pkts/sec, the outgoing rate measured at the server dropped to just $\sim 1,400$ pkts/sec. This constitutes a Packet Rate Reduction of approximately 92.8%. As detailed in Table 6, this efficiency is the result of an average Aggregation Ratio of nearly 14:1, meaning the SAL module successfully buffered and combined, on average, fourteen small non-critical packets into a single larger packet before forwarding.

To confirm the scalability of this mechanism, the P4-TAP Under High Load Scenario increased the incoming traffic volume by a factor of five to $\sim 97,500$ pkts/sec. The SAL module handled this increased load with remarkable consistency, reducing the outgoing packet rate to $\sim 7,000$ pkts/sec. The framework maintained a nearly identical Packet Rate Reduction and Aggregation Ratio. This result demonstrates that the stateful aggregation logic is highly scalable and its efficiency is not degraded by higher traffic volumes, making it a robust solution for optimizing bandwidth in large and active IoT deployments.

5.3 | Discussion

The results from our experiments tell a clear and compelling story. We set out to tackle two of the biggest challenges in modern IoT networks—ensuring that urgent data reaches its destination instantly, and preventing the flood of routine updates from congesting shared network resources. The performance of our P4-TAP framework demonstrates that both goals can be achieved simultaneously, not by adding a more powerful centralized controller, but by making the network’s data plane itself more intelligent.

To properly understand the causes behind the observed performance gains, we conducted an ablation analysis isolating the effect of each module. Using only DSCP marking with strict-priority queuing yielded a latency reduction of approximately 35%, confirming that conventional QoS mechanisms contribute a baseline improvement. However, introducing the In-Data-Plane Traffic Classification (ITC) module and Dynamic Prioritization Engine (DPE) increased the reduction to over 90%, while adding the Stateful Aggregation Logic (SAL) further optimized throughput and queue stability. This layered improvement confirms that the

significant latency drop—up to $94.4 \pm 0.6\%$ under moderate load—is not solely attributable to DSCP priority, but rather to the integrated, stateful decision-making pipeline unique to P4-TAP.

Equally important is how this improvement affects other traffic classes. In many strict-priority systems, low-priority flows risk starvation when high-priority (Expedited Forwarding) packets dominate. To evaluate fairness, we measured the throughput and latency of normal traffic under sustained EF load. Results showed that while critical packets consistently achieved sub-millisecond latency, normal flows maintained 84–88% of their baseline throughput with latency growth limited to $+12 \pm 3$ ms, demonstrating that the SAL’s buffering and timed aggregation effectively mitigates starvation. This balanced resource sharing indicates that P4-TAP’s prioritization strategy remains fair even during heavy critical-traffic periods.

Beyond latency and fairness, the framework’s scalability and efficiency are equally noteworthy. The $92.8 \pm 0.3\%$ packet rate reduction and 14:1 aggregation ratio achieved by the SAL module show that in-data-plane aggregation can significantly reduce header overhead and transmission load without external orchestration. The fact that these gains persisted during a fivefold increase in background traffic highlights the design’s scalability and resource headroom. CPU and queue occupancy profiling further revealed that switch utilization remained below 8%, ensuring that improvements were not the result of underloaded conditions.

Ultimately, our findings validate that empowering the data plane with programmable intelligence can produce a network that is faster for the few (critical data) and more efficient for the many (routine traffic). While these results were obtained using a Mininet-based emulation, they provide strong evidence that data-plane programmability—when combined with adaptive classification and aggregation—can achieve deterministic, low-latency behavior previously limited to hardware-based solutions. Future work will involve evaluating P4-TAP on real switch hardware (e.g., Intel Tofino) under multi-hop and adversarial conditions, and exploring the integration of machine-learning-based policy adaptation to dynamically refine prioritization rules in real time.

6 | Conclusion and Future Work

6.1 | Conclusion

In the bustling world of the Internet of Things, not all data travels at the same speed. We began this work to address a fundamental conflict: the urgent need for critical data to arrive instantly, and the simultaneous need to manage the overwhelming “chatter” of routine updates efficiently. Our solution, the P4-TAP framework, tackled this challenge not by adding more complexity to a central controller, but by embedding intelligence directly into the network itself. By teaching the P4-enabled switch at the edge to act as a smart traffic director, our framework creates a dedicated express lane for critical data and a bulk-shipping lane for everything else.

Our experimental results provide promising evidence of the feasibility of this approach under controlled emulation conditions. Specifically, P4-TAP reduced the end-to-end latency of critical packets by an average of over 94%, ensuring that the most important data gets through with minimal delay. At the same time, our stateful aggregation logic reduced the packet rate from non-critical devices by more than 92%, highlighting the framework’s potential to address both latency and bandwidth inefficiency. Overall, this study demonstrates the viability and strong potential of empowering the network edge with programmability to create smarter and more responsive infrastructures for future IoT environments—pending validation on real hardware and at scale.

6.1 | Future Work

While the results of this study are encouraging, they represent an early step toward building fully intelligent and adaptive IoT networks. A natural continuation of this work involves deploying the P4-TAP framework on real programmable hardware such as the Intel Tofino switch to assess its behavior at true line rates and under the unpredictable conditions of physical networks. Another promising direction is to enhance the traffic

classification mechanism by incorporating machine learning models within the control plane, allowing the framework to learn and refine priority rules dynamically based on long-term traffic behavior.

Future work should also consider strengthening the framework's reliability and observability. Formal verification of the P4 program logic using assertion-based tools could ensure correctness and prevent unintended data-plane behavior. Likewise, integrating in-band network telemetry would enable real-time visibility into latency and throughput, offering valuable insight into performance during operation. Finally, adding runtime monitoring capabilities could allow the system to detect anomalies or performance degradation and adapt its configuration dynamically without the need for redeployment. Together, these enhancements would make P4-TAP a more robust, transparent, and self-optimizing platform for future IoT networking research.

Funding

This research has no funding source.

Conflicts of Interest

The authors declare that there is no conflict of interest in the research.

Ethical Approval

This article does not contain any studies with human participants or animals performed by any of the authors.

Reference

- [1] Gubbi, J., et al., Internet of Things (IoT): A vision, architectural elements, and future directions. *Future generation computer systems*, 2013. 29(7): p. 1645-1660.
- [2] Islam, S.R., et al., The internet of things for health care: a comprehensive survey. *IEEE access*, 2015. 3: p. 678-708.
- [3] Da Xu, L., W. He, and S. Li, Internet of things in industries: A survey. *IEEE Transactions on industrial informatics*, 2014. 10(4): p. 2233-2243.
- [4] Bandyopadhyay, D. and J. Sen, Internet of things: Applications and challenges in technology and standardization. *Wireless personal communications*, 2011. 58(1): p. 49-69.
- [5] Khedr, W.I., A.E. Gouda, and E.R. Mohamed, P4-HLDMC: A novel framework for DDoS and ARP attack detection and mitigation in SD-IoT networks using machine learning, stateful P4, and distributed multi-controller architecture. *Mathematics*, 2023. 11(16): p. 3552.
- [6] Al-Fuqaha, A., et al., Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE communications surveys & tutorials*, 2015. 17(4): p. 2347-2376.
- [7] El-Sayed, A., et al., MP-GUARD: A novel multi-pronged intrusion detection and mitigation framework for scalable SD-IoT networks using cooperative monitoring, ensemble learning, and new P4-extracted feature set. *Computers and Electrical Engineering*, 2024. 118: p. 109484.
- [8] Mekki, K., et al., A comparative study of LPWAN technologies for large-scale IoT deployment. *ICT express*, 2019. 5(1): p. 1-7.
- [9] Kreutz, D., et al., Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 2014. 103(1): p. 14-76.
- [10] Zhang, Z., O. Mara, and K. Argyraki, Network neutrality inference. *ACM SIGCOMM Computer Communication Review*, 2014. 44(4): p. 63-74.
- [11] El-Sayed, A., et al., LBTMA: An integrated P4-enabled framework for optimized traffic management in SD-IoT networks. *Internet of Things*, 2024. 28: p. 101432.
- [12] Bosshart, P., et al., P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review*, 2014. 44(3): p. 87-95.
- [13] Wang, Q., et al. Messages behind the sound: real-time hidden acoustic signal capture with smartphones. in *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*. 2016.
- [14] El-Sayed, A., et al., CO-STOP: A robust P4-powered adaptive framework for comprehensive detection and mitigation of coordinated and multi-faceted attacks in SD-IoT networks. *Computers & Security*, 2025. 151: p. 104349.

-
- [15] Harutyunyan, D. and R. Riggio, How to migrate from operational lte/lte-a networks to e-utran with minimal investment? IEEE Transactions on Network and Service Management, 2018. 15(4): p. 1503-1515.
 - [16] Datta, S. and U. Venkanna, PiGateway: Real-time granular analysis of smart home network traffic using P4. Computer Communications, 2024. 213: p. 309-319.
 - [17] He, W., et al., A p4-based approach to traffic isolation and bandwidth management for 5g network slicing. Tsinghua Science and Technology, 2024. 30(1): p. 171-185.
 - [18] Chen, Y.-F., et al., Adaptive Traffic Control: OpenFlow-Based Prioritization Strategies for Achieving High Quality of Service in Software-Defined Networking. IEEE Transactions on Network and Service Management, 2025.
 - [19] Shahriar, M.S., et al., Prioritized Multi-Tenant Traffic Engineering for Dynamic QoS Provisioning in Autonomous SDN-OpenFlow Edge Networks. arXiv preprint arXiv:2403.15975, 2024.
 - [20] Akem, A.T.J., G. Fraysse, and M. Fiore, Real-Time Encrypted Traffic Classification in Programmable Networks with P4 and Machine Learning. International Journal of Network Management, 2025. 35(1): p. e2320.
 - [21] Oliveira, E.V.d., Automation of traffic generation and testing of programmable networks using P4 and P4Docker. 2025.
 - [22] Chakour, I., et al., Strategic bandwidth allocation for QoS in IoT gateway: Predicting future needs based on IoT device habits. IEEE Access, 2024. 12: p. 6590-6603.
 - [23] Faruqi, N., et al., Cloud iaas optimization using machine vision at the iot edge and the grid sensing algorithm. Sensors, 2024. 24(21): p. 6895.