ARIES: Cost-Efficient Tunnel Gateway Offload on Programmable Switches

Decang Sun*, Gonglong Chen†‡, Chaoliang Zeng*, Kaiqiang Xu*, Han Tian§,
Zili Meng*, Wei Chen¶, Yuxi Wang, and Kai Chen*
*Hong Kong University of Science and Technology, Hong Kong, China
†Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China
‡University of Chinese Academy of Sciences, Beijing, China
§University of Science and Technology of China, Hefei, China
¶Tencent, Shenzhen, China

***Abstract*—Modern interactive applications increasingly rely on transport tunnels to deliver seamless, secure, and low-latency connections across heterogeneous networks. However, implementing these tunnels at edge gateways is costly: connection management, packet encapsulation, reverse-path mapping, and redundancy handling place significant CPU overhead on gateway servers as traffic scale grows. This paper presents ARIES, a hardware-accelerated gateway that shifts the fast path of transport tunnel processing from servers to programmable network switches. By leveraging switch ASICs and optimizing limited on-chip memory, ARIES improves throughput, scalability, and efficiency while reducing the need for expensive server resources. We instantiate ARIES with QUIC/MP-QUIC tunnels as the primary use case, while the core design can be adapted to transport tunnels that expose stable per-session forwarding state. Our Tofino prototype demonstrates that ARIES can support millions of concurrent tunnel sessions and substantially reduce gateway hardware cost compared with server-based tunnel processing.**

***Index Terms*—transport tunnel, programmable switch, QUIC.**

I. INTRODUCTION

As modern applications increasingly demand seamless, reliable, and low-latency connections, operators are moving more transport functionality into edge gateways. QUIC and multi-path QUIC are now widely used by interactive services because they support rapid connection setup, mobility, path diversity, and encrypted transport semantics [15], [19], [7], [39], [9], [22]. QUIC/HTTP tunneling standards and systems further show that transport tunnels are becoming a common substrate for proxying UDP, IP, and application traffic through gateways [29], [28], [25], [37]. However, implementing these tunnels at high-speed gateways brings significant overhead.

Despite advances in transport protocols and tunneling standards, server-based transport tunnel gateways remain costly because of their heavy processing demands. We therefore treat gateway cost as a system-level consequence of repeated software tunnel processing, and scope the later comparison to hardware acquisition cost rather than an unsourced deployment-wide estimate. Core functions, such as tunnel-session management, packet en/decapsulation, reverse-path mapping, multi-path coordination, redundancy handling, and flow matching, place a considerable load on gateway CPUs. As transport-tunnel traffic for interactive services grows, these gateway costs will increase.

979-8-3195-0662-7/26/$31.00 ©2026 IEEE

We emphasize that ARIES targets the *gateway-side* cost of transport tunnels. It does not reduce the CPU overhead of end hosts running QUIC or other transport stacks. Our evaluation therefore focuses on gateway-side cost reduction, not end-host optimization. We use *session record* for complete CPU-side tunnel state, *forwarding context* for reusable gateway/backend/action metadata, *compact field* for its collision-free identifier, and *session lifecycle signal* for validation, update, timeout, or teardown events.

Offloading CPU-intensive processes to programmable network hardware can reduce transport tunnel gateway costs. Programmable switches can manage tunnel fast-path functions, such as session-state lookup, packet en/decapsulation, reverse-path mapping, and optional redundancy filtering, at high throughput and low latency. However, cost-effective tunnel offload on these devices remains challenging. The central challenge is fitting millions of collision-intolerant per-session states into limited ASIC SRAM and pipeline stages: hash compression can create false positives without a universal SYN-like marker, while TCAM prefix merging does not compress exact tunnel-session keys.

Firstly, one category of existing solutions focuses on optimizing stateful forwarding within switch memory. Systems like SilkRoad [21] and Tiara [35] use switch ASICs to accelerate Layer-4 load balancing and connection forwarding, but they are not designed for transport tunnel gateways that must maintain bidirectional tunnel metadata, reverse-path mappings, and optional duplicate filtering over visible redundancy metadata. Secondly, another category expands programmable-switch gateways with additional servers or disaggregated state management. Sailfish [23], FlexGate [26], and recent disaggregated stateful network-function systems [3] improve gateway or network-function scalability, but they either rely on server assistance, target general state placement, or assume traffic distributions that do not directly address ASIC-resident transport tunnel state compression. Recent QUIC tunnel systems optimize transport control, but do not address memory-efficient ASIC-resident tunnel state management at gateways [37].

These systems leave three tunnel-offload challenges: managing tunnel lifecycles and reverse mappings reliably, avoiding hash false positives without TCP-SYN-like markers, and fitting

high-volume tunnel state into constrained ASIC resources.

The key insight behind ARIES is that transport tunnel state has an asymmetric structure: per-session identifiers are high-cardinality, while forwarding context (gateway addresses, backend server pools, tunnel ports, and protocol handlers) is shared across sessions. ARIES keeps complete state in the switch CPU, decomposes repeated forwarding context into compact identifiers, and stores only compact per-session forwarding state in the ASIC fast path. Unlike hash-compression baselines, ARIES uses explicit compact identifiers for repeated forwarding context, avoiding hash-induced false positives in tunnel-state lookup.

In this paper, we propose ARIES, a programmable-switch gateway that offloads transport tunnel fast paths to switch ASICs, enabling cost-efficient support for millions of concurrent tunnel sessions. ARIES targets transport tunnels that expose stable per-session forwarding state, and we instantiate and evaluate the design with QUIC/MP-QUIC tunnels as the primary use case. ARIES separates protocol-specific session identification from switch-resident fast-path forwarding: the CPU handles control packets and table misses for session creation, while compact forward and reverse tables in the ASIC process established tunnel packets, including en/decapsulation, reverse-path lookup, and optional duplicate filtering when visible redundancy metadata is available. Its optimizer is schema-driven rather than QUIC-specific: it uses field widths and observed cardinalities to choose compact table layouts. Our Barefoot Tofino prototype demonstrates high-throughput, low-latency tunnel offload with improved session scalability and hardware cost efficiency.

Our ARIES prototype supports millions of concurrent sessions. Evaluation results indicate that with a single Tofino switch under our testbed configuration, ARIES achieves a hardware cost-efficiency of approximately $22/Gbps, marking a nearly 80% reduction compared to an optimistic commodity-server hardware baseline derived from [38], while maintaining reliable transport tunnel forwarding. In summary, ARIES contributes (i) a collision-free compact-state formulation for tunnel gateways, (ii) a CPU/ASIC split that caches only compact established-session state in the ASIC, and (iii) a Tofino prototype and evaluation of throughput, latency, scalability, memory usage, duplicate filtering, and hardware acquisition cost.

II. Background & Observations

In this section, we first introduce the background of transport tunnels and their workflows, particularly in the context of QUIC/MP-QUIC, multi-path delivery, and gateway tunneling (§II-A). We then discuss programmable switches and previous work on offloading network functions to hardware. Finally, we address the challenges of offloading transport tunnel operations to programmable switches.

A. Transport Tunnel Overview

As modern applications demand higher performance, reliability, and security, edge gateways increasingly terminate or proxy transport tunnels for emerging use cases such as cloud gaming, real-time communication, and interactive streaming. QUIC/MP-QUIC is the primary instantiation used in ARIES because it exposes Connection IDs for session lookup, supports connection migration through control-plane-managed Connection ID updates, and has active multipath extensions for path diversity [15], [8], [7]. These properties make QUIC-style tunnels a strong workload for studying gateway-side tunnel offload. Other protocols, such as TCP/DCTCP (Data Center TCP) [2] or SCTP (Stream Control Transmission Protocol), expose different session identifiers and lifecycle signals. ARIES's state decomposition can apply when such keys are available, but our prototype focuses on QUIC-style transport tunnels.

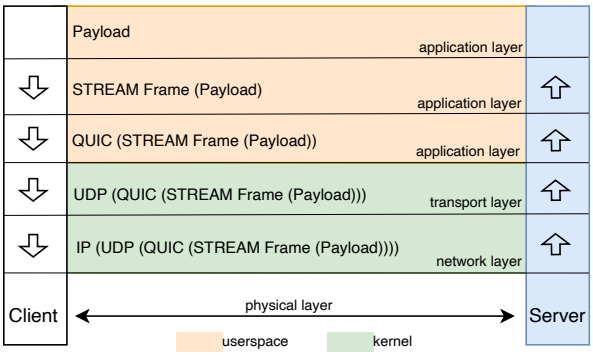


Figure 1. QUIC transport tunnel workflow in the tunnel gateway process.

Transport tunnels maintain session, flow-control, congestion, and encapsulation metadata over flexible underlays such as UDP. They also hide network heterogeneity from backend applications by mapping packets between outer tunnel headers and inner traffic. These stateful mappings create large flow tables in multi-path or encapsulated deployments.

B. Transport Tunnel Workflow

We use QUIC to illustrate the tunnel workflow in ARIES. As shown in Figure 1, the client generates application data, splits it into QUIC streams and frames, and wraps the frames in a QUIC header. The header carries connection-level information, including the Connection ID used to associate packets with a session, before the packet is carried in UDP/IP.

Only a subset of transport-tunnel metadata is suitable for switch fast-path processing. The session identifier used for routing or demultiplexing must be visible to the gateway switch; in QUIC-style tunnels, the Connection ID provides this role. In contrast, transport-internal fields protected by encryption or header protection are treated as opaque by ARIES. Optional functions such as duplicate filtering therefore rely on tunnel-visible metadata explicitly exposed by the tunnel endpoint, such as a redundancy tag in the encapsulation header, rather than on protected transport-internal packet numbers.

Most QUIC implementations run QUIC logic in userspace; the kernel handles only UDP and IP. At the gateway, each tunnel packet requires QUIC header parsing, frame processing, metadata updates, and mapping to inner application traffic.

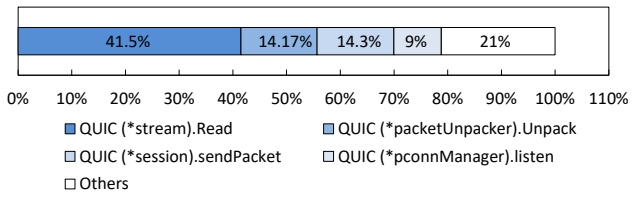


Figure 2. The CPU overhead of a Go-based MP-QUIC stack (gQUIC version 39, multipath per [7]) in the userspace gateway tunnel process.

These operations are modest for one flow but costly across many high-rate tunnel sessions.

C. Motivation: Transport Tunnel Overhead

We profile the userspace tunnel-gateway process implemented with a Go-based MP-QUIC stack (gQUIC version 39, with the multipath extension of [7]) under two transmission rates. This profiling is not intended to represent maximum gateway scale; it isolates the software QUIC tunnel data path that ARIES replaces. The tests use an Intel(R) Xeon(R) Gold 6133 CPU @ 2.50GHz and a Docker container with 2 CPU cores and 4GB memory. We measure CPU utilization and function-level breakdown while replaying multi-path QUIC tunnel traffic. Figure 2 shows the result.

High-rate tunnel. With 100 users, two paths per user, and 3.5 Mbps per user, the total rate reached 363 Mbps over 60.11 seconds. Average process CPU utilization was 156% under Linux CPU accounting, where 100% denotes one fully utilized core. QUIC-related functions accounted for 78.8% of sampled CPU time, dominated by `(*stream).Read` (41.5%), `(*packetUnpacker).Unpack` and `(*session).sendPacket` (around 14% each), and `(*pconnManager).listen` (8.83%).

Low-rate tunnel. With the same 100 users and two paths per user but 1 Kbps per user, the duration remained 60.11 seconds, sampled CPU time was only 620 ms, and average CPU utilization dropped to 1.03% across two cores because the data path carried little traffic after connection setup.

These results show that QUIC tunnel overhead grows with data-plane traffic volume. At high rates, the gateway repeatedly parses headers, unpacks frames, updates session state, and forwards tunnel data [36], [7]. The measurement is taken at the tunnel gateway, not at the end host, and therefore supports the gateway-side offload motivation of ARIES. It is a motivation experiment for the software data path that ARIES replaces, not a maximum-scale server benchmark. This motivates moving established-session processing from gateway CPUs to the programmable-switch fast path.

D. Motivation: Hardware Cost and Background for Programmable Switch

Programmable switches are specialized network devices that process packets through a pipeline of programmable match-action stages. They can execute per-packet operations such as exact-match lookup, metadata update, header rewrite, and forwarding at line rate. These operations match the fast path of a transport tunnel: once a session is established, most packets only need to find the session state, recover tunnel metadata, rewrite headers, and forward the packet. This makes programmable switches attractive for gateway offload.

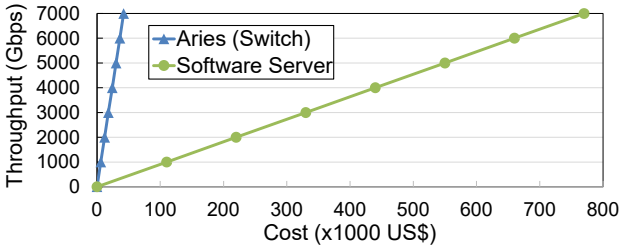


Figure 3. The hardware cost of deploying transport tunnel processing on a programmable switch (ARIES) and software servers. ARIES can reduce the hardware cost by nearly 80%.

Figure 3 compares the hardware cost efficiency of programmable-switch and software-server tunnel processing. Cost efficiency is hardware acquisition cost divided by sustained gateway throughput. We include server/switch hardware cost and exclude operational expenses such as power, rack space, and maintenance. Under this metric, the ARIES prototype achieves higher throughput at approximately 80% lower hardware cost than software servers.

To improve cost-effectiveness and latency, offloading transport tunnel state management to the programmable switch is a viable option. In this context, the state of each session, including connection identifiers, tunnel information, reverse-path metadata, aging state, and packet-processing state, could be maintained in the ASIC. However, managing millions of session entries in the memory-constrained switch is challenging, even with the recent expansion of commodity switch SRAM capacities.

E. Limitations of Prior Hardware Offload

To alleviate storage pressure on the switch, existing work has offloaded network functions to the hardware switch while utilizing software servers to expand the effective state capacity. For example, Sailfish [23] and FlexGate [26] keep many session entries on software servers and use the switch for hot-path processing. These approaches work well when the underlying traffic follows a long-tail distribution, so that a small number of elephant flows hit the hardware data plane and the rest of the mice flows pass through the software server. However, the long-tail distribution does not always hold for unseen traffic, and the traffic distribution can change dynamically [35].

Another method for alleviating storage pressure is using a smaller hash value to identify sessions and compress the session entry size. For example, SilkRoad [21] employs a hashing mechanism for storing the 5-tuple in the switch and providing consistent load-balancer forwarding. Nevertheless, it relies on the first packet (*e.g.*, TCP SYN) to recognize potential hash collisions. For transport tunnel traffic without such packet markers, a hash collision may cause a false positive and attach a packet to the wrong tunnel state. In summary, previous work does not provide a consistent, collision-free, and memory-efficient solution for transport tunnel offloading on programmable switches.

III. ARIES

A. System Model and Protocol Abstraction

ARIES targets transport tunnels that expose four pieces of information: a *session key* for ASIC fast-path lookup; a *session lifecycle signal* for state creation or deletion; *forward and reverse action state* for bidirectional mapping; and optional *sequence state* for duplicate filtering only when the tunnel mode exposes an ASIC-visible monotonic sequence tag.

In our prototype, QUIC/MP-QUIC uses Connection ID as the session key, and the CPU validates Initial or Handshake packets after a table miss. Connection-ID rotation is handled by installing alias keys to the same Full Connection Table record, via control-plane API or explicit CID-mapping signaling between the tunnel endpoint and gateway; packets with an unknown Connection ID still follow the miss path. TCP/DCTCP can use the 5-tuple and SYN or gateway connection events; SCTP can use association tuples or verification tags; registered UDP tunnels can use operator-provided IDs.

ARIES resides at the content-provider gateway: clients send outer tunnel packets, the gateway terminates only the outer tunnel, and unchanged backends receive ordinary inner traffic. Cryptographic transport processing and mutable transport internals remain in software; the ASIC fast path only parses ASIC-visible tunnel metadata and performs steering, header rewrite, reverse mapping, and optional duplicate filtering over a tunnel-visible redundancy tag. The ASIC is a cache of stable forwarding state; unknown keys, control packets, mutable transport internals, and stale entries fall back to the CPU, so missing ASIC entries do not cause incorrect forwarding. Applicability requires a visible session key and lifecycle signal; it maps to QUIC/MP-QUIC (CID and handshake/CID update), TCP/DCTCP (5-tuple and SYN/FIN/RST or gateway events), SCTP (association tuple or verification tag), and registered UDP tunnels (operator-provided ID and registration/timeout). Tunnels whose forwarding-relevant state changes every packet/RTT, or whose keys are not visible, remain outside ARIES's fast-path claim.

B. Overview

ARIES moves the established tunnel fast path from gateway servers to programmable switches. The switch CPU owns complete session records, while the ASIC stores compact fields for ASIC-visible header rewriting, reverse-path mapping, and optional duplicate filtering. The optimizer chooses field decompositions and table layouts from field widths, dependencies, and observed reuse.

C. Architecture

Overall Delivery Pipeline.

In ARIES, client traffic is encapsulated by the transport tunnel stack as usual. The content-provider gateway handles cryptographic tunnel processing in software when needed, while the ASIC fast path rewrites ASIC-visible tunnel headers, forwards inner traffic to backend servers, and uses reverse mappings to encapsulate server responses back into the tunnel.

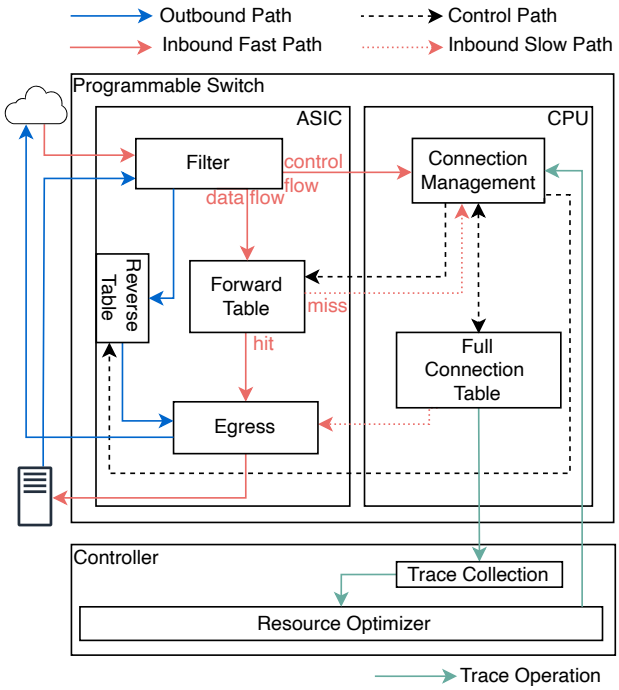


Figure 4. Overview of ARIES.

Figure 4 shows the workflow with a gateway programmable switch and remote controller.

Session Creation and Slow Path.

ARIES does not require the ASIC data plane to recognize a universal first-packet marker. On a Forward Table miss, the packet goes to the CPU, which first checks the Full Connection Table. Existing records indicate aged-out or evicted ASIC entries and are reinstalled; otherwise, the CPU invokes a protocol-specific parser or adapter to validate session creation. After validation or reactivation, the CPU updates the Full Connection Table, selects compact hardware fields, and writes Forward and Reverse Table entries. Later packets from the same session hit the ASIC fast path.

The switch ASIC maintains two tables for established tunnel traffic:

- **Forward Table:** matches user-to-backend packets, recovers tunnel metadata, rewrites ASIC-visible tunnel headers, and forwards the inner packet.
- **Reverse Table:** matches backend-to-user packets and recovers the tunnel identifier and gateway address for encapsulation.

Resource Optimization on the Controller. The Resource Optimizer periodically reads the Full Connection Table and installs field-decomposition/table-layout recommendations into the ASIC Forward, Reverse, and dictionary tables.

IV. HARDWARE OFFLOADING

A. Switch CPU and ASIC Co-design for Transport Tunnel Offload

To support transport tunnels on programmable switches, ARIES keeps complete session state in the CPU and processes

established packets in the ASIC. The CPU maintains the Full Connection Table, whose records include the raw session key, selected backend, forward/reverse action data, aging metadata, and optional visible sequence-tag state. When a session is validated or reactivated, the CPU offloads compact Forward and Reverse Table entries to the ASIC.

The ASIC first extracts the session key and traffic direction with a lightweight header filter. A Forward Table hit copies compact action fields into packet metadata and continues with tunnel header processing; a miss goes to the CPU slow path. Reverse-direction packets use the paired Reverse Table entry so that server responses recover the correct tunnel identifier and gateway address without consulting software.

B. Flow Maintenance with Compression

We present the data structure through three tunnel-gateway requirements and the compression mechanism that realizes them.

Observations. In gateway deployments, many tunnel sessions share gateway IPs, backend server IPs, ports, protocol types, and action metadata. The high-cardinality part is usually the session key, while the forwarding context has much lower cardinality. ARIES exploits this repetition by decomposing shared context into compact identifiers while keeping complete session state in the CPU.

1) Functionality to Implement

Two-way visible-header rewrite. For user-to-backend traffic, the switch rewrites the ASIC-visible tunnel header and forwards the inner packet after software has handled any required cryptographic tunnel processing. For backend-to-user traffic, it applies the corresponding visible tunnel header using reverse-path state.

Destination and Source Conversion. One user session can generate traffic in both directions and across multiple paths. ARIES installs paired Forward and Reverse Table entries: the forward entry maps user-to-server packets to the backend, and the reverse entry maps server-to-user packets back to the tunnel identifier and gateway address.

Deduplication. In multi-path tunnels, the same packet may be transmitted through multiple paths to reduce latency. At the content-provider side, ARIES identifies and eliminates redundant packets when the tunnel mode carries an ASIC-visible monotonic sequence tag, *e.g.*, a redundancy tag carried in the tunnel encapsulation header for selective duplication of latency-sensitive packets. ARIES records the expected tag for each established session and compares later packets against this state to identify duplicates. This mechanism does not require the ASIC to inspect encrypted transport-internal packet numbers; if no visible tag is provided, duplicate filtering is disabled and packets follow the normal forwarding path.

2) Compression

To accommodate as many sessions as possible within limited SRAM, ARIES compresses shared forwarding context through dictionary tables and compact identifiers. A dictionary table stores a repeated field combination once and assigns it a compact identifier; per-session fast-path entries then carry the identifier rather than the original fields.

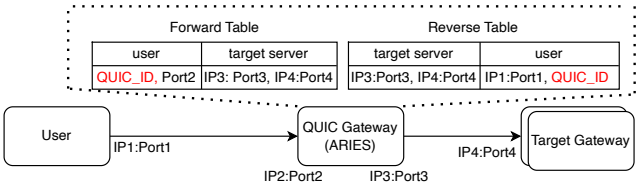


Figure 5. QUIC instantiation of the Forward and Reverse Tables.

By replacing repeated fields with smaller identifiers, ARIES reduces per-session ASIC state. The fast path uses the Forward Table and Reverse Table in Figure 5.

We illustrate the interplay between these tables using QUIC as the instantiated protocol:

1) The Forward Table matches the connection ID (e.g., QUIC Connection ID) and switch port (Port2), with action data including the switch inner IP (IP3), switch inner port (Port3), backend server inner IP (IP4), and backend server inner port (Port4).
2) A packet from the backend server to the user is matched by the Reverse Table, which matches the switch inner IP (IP3), switch inner port (Port3), backend server inner IP (IP4), and backend server inner port (Port4), with action data including the connection ID, switch public IP (IP2), and switch port (Port2).

The matched information is stored in packet metadata for later header processing. The layout itself does not depend on QUIC semantics: QUIC supplies the session key in our prototype, while optional duplicate filtering uses a tunnel-visible redundancy tag rather than QUIC's protected packet number. Other tunnel protocols can use the same layout when they provide the fields in §III-A.

V. Resource Optimizer

ARIES Resource Optimizer (§V-A) decides which repeated fields should be factored into dictionary tables. Its objective is to maximize the number of fast-path sessions that fit in ASIC SRAM under stage and RAM constraints. The optimizer operates on field widths and observed value cardinalities, not on QUIC semantics; QUIC only determines which fields appear in the session schema of our prototype. Meanwhile, we design an aging mechanism (§V-B) for table entries to clean up the switch's memory. Active-standby failover is discussed as a deployment extension in §V-C.

A. Modeling

Given recent traffic information from the Full Connection Table, the optimizer chooses which field combinations should become dictionary entries and how much SRAM should be reserved for them. The optimizer runs in the control plane and updates table layouts periodically; it is not on the per-packet fast path. This formulation is schema-driven: the same variables can represent QUIC Connection IDs, five-tuples, or other tunnel keys once the parser exposes the corresponding

fields. We formulate the problem as a mixed integer non-linear optimization problem in Figure 6, which can be readily solved by off-the-shelf solvers such as Solving Constraint Integer Programming (SCIP) [4].

Decision Variables. The optimizer operates on the selected field combinations for the Forward Table and Reverse Table, respectively. Constraints Eqs. (1h), (1i), and (1j) clarify the types of decision variables.

Objective Function. The objective function in Eq. 1a maximizes the number of entries that can be stored in the ASIC fast path, where the optimal solution $X^* = (\mathbf{x}_i, \mathbf{x}')$ represents the selected layout. The objective is further expressed in Eq. 1f, which gives a continuous capacity estimate; the installed number of entries is the integer floor of this value after the layout is chosen. It is the available bit capacity ($c(s_{all} - s_1 - s_2)$) over the required bits for one entry ($b_1 + b_2 + O + f$). Here, b_1 and b_2 denote the bit widths of the compact identifiers selected for the Forward Table and Reverse Table dictionaries, O denotes the bits of fields that are not factored into dictionaries, and f denotes fixed per-entry metadata bits. s_1 and s_2 represent the stages consumed by the Forward Table and Reverse Table dictionaries, respectively.

Consistent Constraints. Constraints Eq. 1b and Eq. 1c secure that any field is considered at most once, where binary parameters b_{ij}, b'_{ij} denote the inclusion of field j in field combination i.

Resource Constraints. The constraint Eq. 1g ensures that the stored entries do not exceed the MAP RAM limit, where s_1 and s_2 denote the stage usage of the Forward Table and Reverse Table dictionaries, respectively. $mRAM$ denotes the total MAP RAM capacity available to the fast path, and $mRAM_{pe}$ denotes the MAP RAM consumed per fast-path entry. s_1 and s_2 are further calculated by dividing the total number of bits for the Forward Table and Reverse Table dictionary entries by the bit capacity of each stage (Eq. 1d). Take s_1 as an example, one entry is composed of one optimal field combination and corresponding original fields, requiring $\lceil \log_2 a_i \rceil$, $\sum_j b_{ij} o_j$ bits, respectively. a_i, o_j, c denote the number of different value instances of the field combination i that occurred in the last time interval, bits for the original field j, and the SRAM capacity of each stage, respectively. To solve the Decomposition Optimizer model, we apply the SCIP solver. We linearize some nonlinear constraints for the tractability of solving our non-linear optimizations.

$$\mathcal{C}^* = \max\ C \tag{1a}$$

$$\text{s.t.} \sum_i x_i b_{ij} \quad \leq 1, \quad \forall j = 1, ..., |F|, \tag{1b}$$

$$\sum_i x'_i b'_{ij} \quad \leq 1, \quad \forall j = 1, ..., |F'|, \tag{1c}$$

$$s_1 \quad = \lceil \frac{\sum_i x_i a_i (\lceil \log_2 a_i \rceil + \sum_j b_{ij} o_j)}{c} \rceil, \tag{1d}$$

$$s_2 \quad = \lceil \frac{\sum_i x'_i a'_i (\lceil \log_2 a'_i \rceil + \sum_j b'_{ij} o'_j)}{c} \rceil, \tag{1e}$$

$$C \quad = \frac{c(s_{all} - s_1 - s_2)}{b_1 + b_2 + O + f}, \tag{1f}$$

$$C \quad \leq \frac{mRAM(s_{all} - s_1 - s_2)}{mRAM_{pe}}, \tag{1g}$$

$$x_i, x'_i \quad = \{0, 1\}, \quad \forall i = 1, ..., |R|, \forall i = 1, 2, 3, \tag{1h}$$

$$s_1, s_2, b_1, b_2 \quad = \{0, 1, 2, 3, ...\}, \tag{1i}$$

$$C \quad \geq 0. \tag{1j}$$

Figure 6. The non-linear programming formulation for Resource Optimizer.

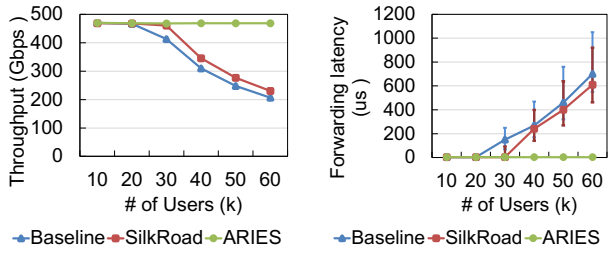
B. Efficient Entry Aging

In the context of switch ASICs, traditional entry aging methods that use countdown timers refreshed on entry hits cannot be applied due to hardware constraints, which only allow entry-aged notifications and not entry-hit notifications [40]. The Tofino *idle_timeout* counter is reset by a table hit and emits a notification only after the entry has been idle for T_{age}. Therefore, if a notification arrives at time t_n, ARIES infers the last hit time as $t_{hit} = t_n - T_{age}$. The Forward Table entry in the ASIC includes a matching field, action data, and aging time, while the Full Connection Table in the CPU stores the raw session key, tunnel action data, and last hit time in a sorted list. Entry aging synchronization updates the last hit time only when both directions of a session exceed the aging timeout. Session management then creates or offloads entries according to available Forward Table space.

C. Seamless Upgrade

Active-standby failover is orthogonal to ARIES's fast-path design. Our current prototype does not implement failover; we discuss how standard active-standby deployment can be integrated with ARIES as a deployment extension. Existing thermal upgrade schemes reduce outage time but require new chip architectures and still result in a cut-off time of 30-50 microseconds [12]. In a production deployment, ARIES can follow the common active-standby model for stateful gateways: time-sensitive events such as entry insertion or deletion are synchronized immediately, while less time-sensitive metadata such as entry aging time is synchronized periodically. During an upgrade, the standby gateway can be upgraded first, traffic can shift to the upgraded standby, and the primary can then be upgraded before traffic shifts back [12].

The 200K entries/s Runtime API figure (§VI-A) is an API-level upper bound, corresponding to an optimistic lower bound of about 10 seconds for 2M entry operations. In contrast, the end-to-end Ruijie 9500 update rate measured in §VII-F gives about 40–45 minutes to rebuild a 2M-entry ASIC cache. During reconstruction, packets whose entries have not yet been restored follow the CPU slow path; depending on reinstallation order, a large active flow may remain on the slow path for nearly the full reconstruction interval. Active flows are not removed by idle aging.



(a) Throughput. (b) Forwarding latency.

Figure 7. Performance evaluation under cloud gaming scenario simulated on the testbed.

VI. Implementation

A. Implementation Prototype

We built a P4 prototype of ARIES on switch.p4 and compiled it on a programmable switch ASIC. The baseline switch.p4 implements various network functions required by a typical cloud data center in approximately 5,000 lines of P4 code. A simplified version of the baseline switch.p4 is open sourced in [1]. We added about 450 lines of P4 code to implement all tables and metadata required by ARIES.

Data Plane on Switch ASIC. We implement all tables as exact-match tables, including the Forward Table, Reverse Table, and dictionary tables. For each entry in the Forward Table, we utilize the extern feature *idle_timeout* of the Tofino chip to allow the hardware to perform an aging countdown. When the countdown of a certain entry ends, the switch ASIC chip will send an aging notification message to the control plane (running on the switch CPU) so that the control plane can update the aging time of the corresponding entry. In order to identify duplicate packets in tunnel modes that expose redundancy metadata, we use the *direct register* for each entry to record the next visible sequence tag in the flow.

Control Plane on Switch CPU. We also implement a control plane in the switch software for Connection Management. This includes managing connection state and adding, deleting, modifying, and querying table entries along with the aging time updating of entries. To achieve this, the control plane employs a framework based on the Runtime API of Tofino. The Runtime API is preferred over gRPC (Google Remote Procedure Call) communication due to its higher efficiency, which is crucial for rapidly updating table entries. Prior work reports an API-level throughput of approximately 200K table operations/s using the Tofino Runtime API [35]; our end-to-end measurement on the Ruijie 9500 in §VII-F is substantially lower. On a miss, the CPU validates or finds the session record, installs any needed dictionary entry first, then installs Forward/Reverse entries; deletion uses the reverse order so the ASIC never keeps a dangling compact identifier. CID rotation is handled by installing an alias key to the same CPU record, and packets during updates either hit a complete entry or take the miss path.

B. Testbed Deployment

For evaluating the performance of ARIES, we set up a testbed using a hardware traffic generator directly connected to our system.

We leverage a high-performance, multi-functional hardware traffic generator to create synthetic transport tunnel flows, which are sent to ARIES and then routed back to the generator. The tester generates multiple paths for each emulated session by varying link profiles, and in the duplicate-filtering experiment it sends redundant packets with the same tunnel-visible sequence tag over different paths.

VII. Evaluation

We evaluate ARIES using testbed experiments against a full-entry exact-match baseline and a SilkRoad-style hash-compressed layout [21]; other gateway/L4-load-balancing systems are discussed qualitatively in §VIII. We answer four questions: (1) Can ARIES sustain high-rate gateway forwarding? (2) How many sessions fit in ASIC memory? (3) How much does state decomposition improve memory efficiency? (4) What benefit does duplicate filtering provide under redundant multi-path traffic?

Parameter Settings. The baseline approach uses one generic stateful table keyed by the five-tuple. The action data contains all fields needed to construct the tunnel, including the 8-byte Connection ID, outer IP/UDP addresses and ports, 4-byte flow id, and optional 4-byte visible sequence tag for duplicate filtering.

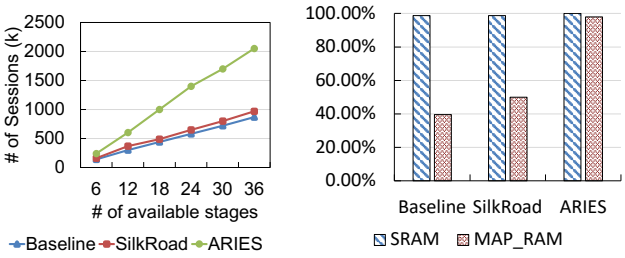
SilkRoad-style layout [21] uses hashing to compress match fields, which can introduce false positives under hash conflicts. A false positive forwards traffic that should not match the entry, causing packet loss or latency jitter. SilkRoad resolves some conflicts using TCP SYN; for tunnel-state comparison, we keep the same tunnel action fields as Baseline and apply only its hash-compressed match layout with a 4-byte compressed field (around 0.01% false positives for 60K users with 1.8M sessions).

ARIES continuously optimizes table item design by loading application traffic into the optimization model.

Large Scale Settings. We utilize high-performance Spirent testers [30] to simulate concurrent sessions generated by large-scale users. Each emulated user generates 30 flows, so the 10K–60K user range corresponds to 0.3M–1.8M concurrent tunnel sessions. The ARIES prototype runs on a Ruijie 9500 Tofino switch connected to the Spirent tester through five 100GbE ports. The 470 Gbps result is the maximum offered load under this configuration with 512-byte tunnel packets; Ethernet framing overhead accounts for the gap to the 500 Gbps aggregate line rate.

A. Overall Performance

As shown in Figure 7(a), ARIES preserves the highest measured throughput almost unchanged at around 470 Gbps when the number of users increases from 10K to 60K. Therefore, this experiment stresses whether increasing tunnel-session state



(a) Session scalability. (b) RAM utilization.

Figure 8. The session scalability comparison.

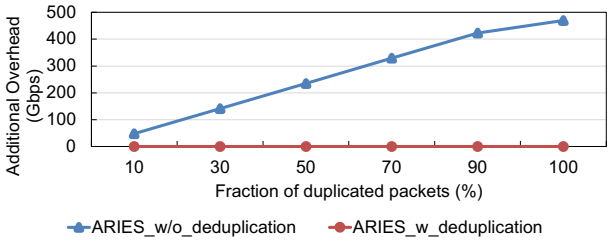


Figure 9. The efficiency of duplicate filtering.

pushes traffic out of the ASIC fast path, rather than the raw packet-processing limit of the chip. We report hardware cost efficiency as hardware acquisition cost divided by sustained gateway throughput under our testbed configuration. For a consistent normalized comparison, we adopt the 2018 hardware-cost figures reported in [38] for both platforms: $10,500 for the programmable switch and $4,400 for a commodity server with a 40GbE NIC. Applying our measured 470 Gbps gateway throughput to the reported switch cost gives $22/Gbps. For the server baseline, we optimistically assume that a $4,400 commodity server with a 40GbE NIC sustains full NIC line rate, yielding $110/Gbps; matching 470 Gbps therefore requires twelve servers ($52,800). Specifically, ARIES achieves 2.28× and 2.04× throughput improvements over Baseline and SilkRoad-style hash compression, respectively, when the number of users reaches 60K.

When the user scale exceeds 30,000, the session scale exceeds 0.9M. This exceeds the fast-path capacities of Baseline (0.79M) and SilkRoad-style hash compression (0.88M). The additional sessions are handled by the slow path, so the slow-path fraction increases with user count, reducing aggregate throughput and increasing forwarding latency. For ARIES, nearly 2M sessions stay in on-chip memory, so traffic remains on the ASIC fast path.

B. Scalability

As shown in Figure 8, ARIES achieves highest session scalability consistently. Specifically, ARIES improves the session scalability by 2.36× and 2.11× compared with Baseline and SilkRoad-style hash compression, when the available logical stage budget is 36. We use logical stage budget to sweep the memory-placement constraint across the forward, reverse, and dictionary tables; the deployed prototype maps this budget onto the available ingress/egress pipeline resources of the switch.

Figure 8(b) shows that ARIES achieves nearly the maximum resource utilization to store sessions, while Baseline and SilkRoad-style hash compression are SRAM-limited even when MAPRAM remains available.

C. Benefits of Duplicate Filtering

In this experiment, the Spirent tester emulates redundant multi-path traffic with two paths per session. For each selected data packet, the tester sends two tunnel packets carrying the same payload and tunnel-visible sequence tag through different paths, emulating a redundancy mode used to reduce tail latency under heterogeneous links. ARIES keeps the latest visible sequence state for each established session and drops the redundant copy at the gateway. Deduplication is optional and only applies to tunnel modes that expose monotonic packet or sequence identifiers to the ASIC. Figure 9 shows that ARIES can reduce the additional throughput by nearly half compared with forwarding redundant packets to the backend.

D. Supporting Heterogeneous Traffic

To evaluate the performance of ARIES under heterogeneous traffic models, we emulated four typical applications with different proportion of TCP traffic, *i.e.*, web surfing (around 98% TCP flow) [33], cloud gaming (around 20% TCP flow for gaming connections) [6], live streaming (about 25% TCP flow for user login) [31] and online meeting (about 40% TCP flow) [16].

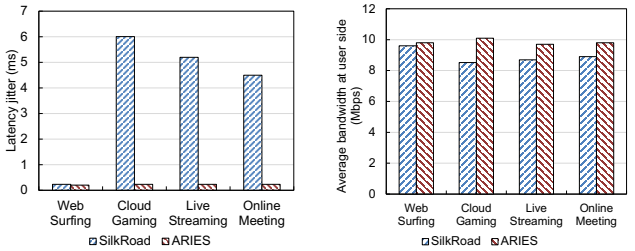
To obtain the latency jitter and bandwidth from the client perspective, we emulate 60,000 users concurrently generating 1.8 million different flows in the Spirent tester [30]. If the client does not receive the desired data within 3 seconds of timeout and retransmission [27], the source port in the 5-tuple will be randomly changed to alleviate the session conflict problem of UDP traffic in the SilkRoad-style hash-compressed layout. Latency jitter and bandwidth data are averages from 10 minutes of testing.

Performance Across Various Scenarios: Figure 10 shows that ARIES keeps the bandwidth around 10Mbps and has the lowest latency jitter because more sessions remain in the ASIC fast path.

Bandwidth Performance with Multiple Applications: Figure 12 shows the client-side bandwidth when web surfing and cloud gaming traffic run at the same time. ARIES keeps bandwidth close to 10Mbps across the run. SilkRoad-style hash compression shows periodic drops because unresolved UDP hash conflicts send some packets to the wrong session, causing frame loss at the client.

E. Optimizer Convergence

Figure 11 shows that the optimizer quickly stabilizes once recent tunnel-session records are available. In our workload, a small trace already exposes the repeated field combinations used by the gateway, so the selected layout supports around 1.8M sessions. Feeding substantially more records adds rare combinations and can reserve extra dictionary space, so we



(a) Latency jitter. (b) Average bandwidth.

Figure 10. Performance under various scenarios from clients' perspective.

use recent bounded samples rather than the entire historical table.

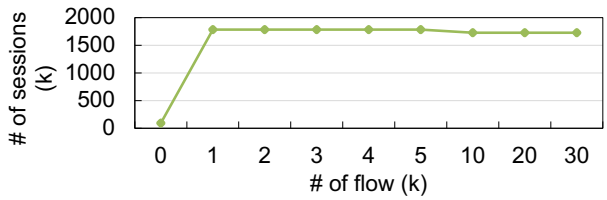


Figure 11. Optimizer convergence.

F. Control-Plane Scalability and Session Churn

The 200K entries/s Runtime API number is an API-level upper bound. We therefore use a representative end-to-end table-update microbenchmark on the Ruijie 9500 switch CPU (Intel Xeon D-1527, 4 cores, 16GB memory). As Table I shows, the 32-bit-key/32-bit-action exact-match table sustains 761–790 insertions/s and 795–847 deletions/s across tested batch sizes; LPM updates are lower (600–650 adds/s) and are not on the session fast path. Since one ARIES session requires at least one Forward and one Reverse entry, these measurements imply an optimistic upper bound of 380–395 session installations/s. Under balanced insertion/deletion churn, the corresponding upper bound is approximately 190 sessions/s, before dictionary updates and ARIES-specific software bookkeeping. Beyond this bound, the update backlog grows and the control plane becomes the bottleneck. Let p_s denote the packet-weighted fraction of traffic whose sessions are waiting for installation or reinstallation. The fast-path hit ratio is $1 - p_s$, and the slow-path penalty is the corresponding throughput reduction and latency increase relative to the ASIC fast path. The value of p_s increases with session arrival/departure rate, short-flow ratio, table-miss rate, and CID-rotation frequency because they share the same update budget. Active large flows are not removed by idle aging; missing entries use the slow path while being reinstalled. The prototype does not implement failover; active-standby synchronization of the CPU Full Connection Table is possible, but at the measured exact-match update rate, reinstalling a 2M-entry ASIC cache requires at least about 40–45 minutes, excluding dictionary updates and ARIES-specific bookkeeping.

Metric	Capacity
Exact-match insertion	761–790 entries/s
Exact-match deletion	795–847 entries/s
Paired Forward/Reverse installation	380–395 sessions/s
Balanced insertion/deletion churn	≈190 sessions/s

Table I
END-TO-END CONTROL-PLANE UPDATE CAPACITY ON THE RUIJIE 9500 SWITCH CPU.

Resource	Baseline	SilkRoad-style	ARIES
Match Crossbar	9.79%	4.44%	4.20%
SRAM	66.91%	54%	24.92%
MAP RAM	26.91%	26.82%	26.38%
TCAM	0%	0%	0%
VLIW Action	3.73%	2.87%	1.75%
Hash Bits	13.15%	11.17%	10.21%
Stateful ALUs	17.18%	14.58%	7.29%
Packet Header Vector	12.57%	9.10%	9.90%

Table II
ADDITIONAL H/W RESOURCES USED BY THE BASELINE APPROACH, SILKROAD-STYLE HASH COMPRESSION AND ARIES WITH 800K TUNNEL SESSIONS, NORMALIZED BY THE USAGE OF THE BASELINE SWITCH.P4.

G. Memory Efficiency

ASIC Resource Consumption.

Table II shows that ARIES saves 45.3% and 36.5% memory overhead compared with Baseline and SilkRoad-style hash compression, respectively. For ARIES, the additional resource usage is less than 30% for all listed hardware resources. Baseline and SilkRoad-style hash compression consume much more SRAM (66.91% and 54%, respectively).

Match Crossbar. The exact-match tables mainly consume match crossbar resources, and the usage grows with match-key width. The match field length of ARIES (4B) and SilkRoad-style hash compression (5B) is much smaller than that of Baseline (13B), so both use less crossbar than Baseline. Despite having a session key length that is one byte shorter than ARIES, SilkRoad-style hash compression uses more stages due to its lower compression efficiency. Specifically, it consumes 23 stages compared to ARIES's 16 stages when handling 800k tunnel sessions.

Hash bits. The hash operations in exact-match tables and the multi-way hash addressing of Bloom filters consume additional hash bits. The number of hash bits is also related to key length. Tofino stores exact-match keys using Cuckoo hashing, so longer keys require more hash bits to resolve potential conflicts. Both ARIES and SilkRoad-style hash compression have smaller key lengths than Baseline.

SRAM. For SRAM, ARIES has the smallest fast-path entry size because repeated forwarding fields are factored into dictionary entries. As a result, ARIES uses the least SRAM (24.92%).

MAP RAM. It mainly stores per-session state such as the visible sequence tag and aging timeout. Each active session requires corresponding state, so the three schemes have similar MAP RAM overhead at the same session scale.

TCAM. None of the three schemes uses LPM or ternary match, so TCAM overhead is 0%.

VLIW Actions. The VLIW (very long instruction word) actions are used for packet modifications [5]. This resource is consumed per stage, and ARIES uses the fewest stages, so its ratio is the smallest.

The duplicate-filtering implementation uses stateful ALUs

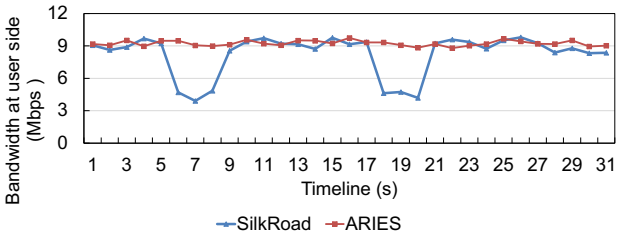


Figure 12. Bandwidth timeline when supporting two applications (web surfing and cloud gaming) traffic simultaneously.

to perform transactional read-check-update, and it is required in each stage. Therefore, ARIES consumes the least Stateful ALUs.

All three schemes define a few metadata fields to carry session-key information across tables. Note that ARIES uses slightly more PHV (Packet Header Vector) bits than SilkRoad-style hash compression in the current implementation. ARIES passes more compact identifiers (*i.e.*, srcid and dstid), but the ratio remains small (9.9%).

VIII. Related Work

Switch-based load balancing and stateful forwarding. Software and hardware load balancers such as Ananta [24], Duet [13], Maglev [10], SilkRoad [21], and Tiara [35] provide scalable Layer-4 forwarding and connection-consistent load balancing. SilkRoad motivates the hash-compressed ASIC-resident baseline we reproduce, and Tiara further improves cache/server coordination for stateful L4 load balancing. These systems are complementary to ARIES: their published designs focus on L4 forwarding state, while ARIES specializes the ASIC-resident state layout for bidirectional tunnel metadata, reverse tunnel action state, and optional gateway-side duplicate filtering.

Programmable-switch gateways and network-function offload. Sailfish [23] and FlexGate [26] use programmable switches to accelerate gateway functions, often relying on software servers or traffic-distribution assumptions to handle state that does not fit in the switch. Disaggregated stateful network functions [3] study how to place and manage NF state across devices. P4-IPsec [14] and the transparent broadband VPN gateway [32] accelerate VPN/tunnel processing on switching hardware, but target aggregate tunnels rather than compression of millions of collision-intolerant per-session states. These systems address gateway/NF scalability and state placement, while ARIES focuses on keeping the established transport-tunnel fast path in the ASIC through decomposition of repeated tunnel state.

SmartNIC and reconfigurable hardware offload. SmartNIC and FPGA-based systems such as FlexNIC [17], ClickNP [20], and HyperNAT [11] offload packet processing to programmable NICs or reconfigurable hardware. These platforms provide more flexible compute than switch ASICs, but they sit on the server path and do not provide the same in-network line-rate forwarding model. ARIES instead targets the established-session fast path on gateway switches.

Fault-tolerant in-switch state. RedPlane [18] studies fault-tolerant stateful applications on programmable switches. This direction is complementary to ARIES. Our prototype does not evaluate failover; as discussed in §V-C, active-standby deployment can be integrated as an operational extension without changing the fast-path design.

QUIC and multipath transport optimization. MP-QUIC [7], [8], Xlink [39], CellFusion [22], Converge [9], and TECC [37] improve multipath transport, application QoE, or QUIC tunnel control. These works motivate our workload, but they do not offload gateway tunnel state to programmable switch ASICs. ARIES is complementary to such tunnel-control systems, including TECC: their control logic can choose paths and expose tunnel metadata, while ARIES keeps the established-session forwarding state in the switch fast path.

IX. Discussion and Future Work

ARIES avoids FPGA-based SRAM expansion because it adds ASIC–FPGA round-trip latency to the established-session path; the ASIC-resident layout avoids this off-chip memory round trip and keeps established-session processing within the switch pipeline. ASIC entries are only a cache of the CPU Full Connection Table, so misses from aging can be reinstalled without changing correctness. We report hardware cost rather than full energy; larger deployments can shard sessions across gateway switches with control-plane state replication [18].

X. Conclusion

This paper presented ARIES, a programmable-switch gateway for cost-efficient transport tunnel offload. ARIES targets transport tunnels that expose stable per-session forwarding state, with QUIC/MP-QUIC as the primary instantiation in our prototype. By keeping complete session state in the switch CPU and compact forward/reverse state in the ASIC, ARIES offloads established tunnel sessions while respecting on-chip memory and pipeline constraints. Our evaluation shows that this design improves throughput, session scalability, memory efficiency, and gateway hardware cost compared with server-based tunnel processing and ASIC-resident hash compression.

Acknowledgment

We thank the anonymous ICNP reviewers and our shepherd for their insightful comments. This work is supported by the National Key R&D Program of China (No. 2025YFE0204100), the Science and Technology Development Fund of Macao S.A.R. (FDCT) under grant No. 0074/2025/AMJ, the Guangdong Basic and Applied Basic Research Foundation (Nos. 2026A1515011754 and 2023B1515130002), the Key Research and Development and Technology Transfer Program of Inner Mongolia Autonomous Region (No. 2025YFHH0110), the Shenzhen Basic Research Program (Nos. JCYJ20250604183035046 and KJZD20230923113800001), the Hong Kong RGC TRS T41-603/20R, and TACC [34]. Kai Chen is the corresponding author.

REFERENCES

[1] switch.p4. Online: https://github.com/p4lang/switch, 2020.

[2] Mohammad Alizadeh, Albert Greenberg, David A Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. Data center tcp (dctcp). In *Proceedings of the ACM SIGCOMM 2010 Conference*, pages 63–74, 2010.

[3] Deepak Bansal, Gerald DeGrace, Rishabh Tewari, Michal Zygmunt, Silvano Gai, Mario Baldi, Krishna Doddapaneni, Arun Selvarajan, Arunkumar Arumugam, Balakrishnan Raman, Avijit Gupta, Sachin Jain, Deven Jagasia, Evan Langlais, Pranjal Srivastava, Rishiraj Hazarika, Neeraj Motwani, Soumya Tiwari, Stewart Grant, Ranveer Chandra, and Srikanth Kandula. Disaggregating stateful network functions. In *Proceedings of Networked Systems Design and Implementation (NSDI)*. USENIX, 2023.

[4] Ksenia Bestuzheva, Mathieu Besançon, Wei-Kun Chen, Antonia Chmiela, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Oliver Gaul, Gerald Gamrath, Ambros Gleixner, et al. The SCIP Optimization Suite 8.0. *arXiv preprint arXiv:2112.08872*, 2021.

[5] Pat Bosshart, Glen Gibb, Hun-Seok Kim, George Varghese, Nick McKeown, Martin Izzard, Fernando Mujica, and Mark Horowitz. Forwarding metamorphosis: Fast programmable match-action processing in hardware for sdn. *SIGCOMM Comput. Commun. Rev.*, 43(4):99–110, aug 2013.

[6] Marc Carrascosa and Boris Bellalta. Cloud-gaming: Analysis of google stadia traffic. *Computer Communications*, 188:99–116, 2022.

[7] Quentin De Coninck and Olivier Bonaventure. Multipath quic: Design and evaluation. In *Proceedings of the 13th international conference on emerging networking experiments and technologies*, pages 160–166, 2017.

[8] Quentin De Coninck and Olivier Bonaventure. Multipath extensions for quic (mp-quic). *IETF Internet-Draft draft-decor-incк-quic-multipath-06*, 2020.

[9] Sandesh Dhawaskar Sathyanarayana, Kyunghan Lee, Dirk Grunwald, and Sangtae Ha. Converge: Qoe-driven multipath video conferencing over webrtc. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 637–653, 2023.

[10] Daniel E Eisenbud, Cheng Yi, Carlo Contavalli, Cody Smith, Roman Kononov, Eric Mann-Hielscher, Ardas Cilingiroglu, Bin Cheyney, Wentao Shang, and Jinnah Dylan Hosein. Maglev: A fast and reliable software network load balancer. In *Nsdi*, volume 16, pages 523–535, 2016.

[11] Shaoke Fang, Qingsong Liu, and Wenfei Wu. Hypernat: Scaling up network address translation with smartnics for clouds. In *2021 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2021.

[12] Yong Feng, Zhikang Chen, Haoyu Song, Wenquan Xu, Jiahao Li, Zijian Zhang, Tong Yun, Ying Wan, and Bin Liu. Enabling in-situ programmability in network data plane: From architecture to language. In *Proc. of USENIX NSDI*, 2022.

[13] Rohan Gandhi, Hongqiang Harry Liu, Y Charlie Hu, Guohan Lu, Jitendra Padhye, Lihua Yuan, and Ming Zhang. Duet: Cloud scale load balancing with hardware and software. *ACM SIGCOMM Computer Communication Review*, 44(4):27–38, 2014.

[14] Frederik Hauser, Marco Häberle, Mark Schmidt, and Michael Menth. P4-ipsec: Site-to-site and host-to-site vpn with ipsec in p4-based sdn. *IEEE Access*, 8:139567–139586, 2020.

[15] Jana Iyengar, Martin Thomson, et al. Quic: A udp-based multiplexed and secure transport. In *RFC 9000*. Internet Engineering Task Force (IETF) Fremont, CA, USA, 2021.

[16] Mehdi Karamollahi and Carey Williamson. Simulation modeling of zoom network traffic. In *2022 30th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 57–64. IEEE, 2022.

[17] Antoine Kaufmann, SImon Peter, Naveen Kr Sharma, Thomas Anderson, and Arvind Krishnamurthy. High performance packet processing with flexnic. In *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 67–81, 2016.

[18] Daehyeok Kim, Jacob Nelson, Dan R. K. Ports, Vyas Sekar, and Srinivasan Seshan. Redplane: Enabling fault-tolerant stateful in-switch applications. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. Association for Computing Machinery, 2021.

[19] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, et al. The quic transport protocol: Design and internet-scale deployment. In *Proceedings of the conference of the ACM special interest group on data communication*, pages 183–196, 2017.

[20] Bojie Li, Kun Tan, Layong Luo, Yanqing Peng, Renqian Luo, Ningyi Xu, Yongqiang Xiong, Peng Cheng, and Enhong Chen. Clicknp: Highly flexible and high performance network processing with reconfigurable hardware. In *Proceedings of the 2016 ACM SIGCOMM Conference*, pages 1–14, 2016.

[21] Rui Miao, Hongyi Zeng, Changhoon Kim, Jeongkeun Lee, and Minlan Yu. Silkroad: Making stateful layer-4 load balancing fast and cheap using switching asics. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 15–28, 2017.

[22] Yunzhe Ni, Zhilong Zheng, Xianshang Lin, Fengyu Gao, Xuan Zeng, Yirui Liu, Tao Xu, Hua Wang, Zhidong Zhang, Senlang Du, et al. Cellfusion: Multipath vehicle-to-cloud video streaming with network coding in the wild. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 668–683, 2023.

[23] Tian Pan, Nianbing Yu, Chenhao Jia, Jianwen Pi, Liang Xu, Yisong Qiao, Zhiguo Li, Kun Liu, Jie Lu, Jianyuan Lu, et al. Sailfish: Accelerating cloud-scale multi-tenant multi-service gateways with programmable switches. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, pages 194–206, 2021.

[24] Parveen Patel, Deepak Bansal, Lihua Yuan, Ashwin Murthy, Albert Greenberg, David A Maltz, Randy Kern, Hemant Kumar, Marios Zikos, Hongyu Wu, et al. Ananta: Cloud scale load balancing. *ACM SIGCOMM Computer Communication Review*, 43(4):207–218, 2013.

[25] Tommy Pauly, David Schinazi, Alex Chernyakhovsky, Mirja Kuehlewind, and Magnus Westerlund. Proxying IP in HTTP. RFC 9484, October 2023.

[26] Kun Qian, Sai Ma, Mao Miao, Jianyuan Lu, Tong Zhang, Peilong Wang, Chenghao Sun, and Fengyuan Ren. Flexgate: High-performance heterogeneous gateway in data centers. In *Proceedings of the 3rd Asia-Pacific Workshop on Networking 2019*, pages 36–42, 2019.

[27] Pasi Sarolahti, Markku Kojo, and Kimmo Raatikainen. F-rto: an enhanced recovery algorithm for tcp retransmission timeouts. *ACM SIGCOMM Computer Communication Review*, 33(2):51–63, 2003.

[28] David Schinazi. Proxying UDP in HTTP. RFC 9298, August 2022.

[29] David Schinazi and Lucas Pardue. HTTP Datagrams and the Capsule Protocol. RFC 9297, August 2022.

[30] Spirent. Unlock the potential of new technology. Online: https://www.spirent.com, 2023.

[31] Kunwadee Sripanidkulchai, Bruce Maggs, and Hui Zhang. An analysis of live streaming workloads on the internet. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 41–54, 2004.

[32] Kenji Tanaka, Takashi Uchida, Yuki Matsuda, Yuki Arikawa, Shinya Kaji, and Takeshi Sakamoto. Transparent broadband vpn gateway: Achieving 0.39 tbps per tunnel with bump-in-the-wire. In *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications*, pages 2318–2327, 2024.

[33] Guowu Xie, Marios Iliofotou, Thomas Karagiannis, Michalis Faloutsos, and Yaohui Jin. Resurf: Reconstructing web-surfing activity from network traffic. In *2013 IFIP Networking Conference*, pages 1–9. IEEE, 2013.

[34] Kaiqiang Xu, Decang Sun, Hao Wang, Zhenghang Ren, Xinchen Wan, Xudong Liao, Zilong Wang, Junxue Zhang, and Kai Chen. Design and operation of shared machine learning clusters on campus. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, pages 295–310, 2025.

[35] Chaoliang Zeng, Layong Luo, Teng Zhang, Zilong Wang, Luyang Li, Wenchen Han, Nan Chen, Lebing Wan, Lichao Liu, Zhipeng Ding, et al. Tiara: A scalable and efficient hardware acceleration architecture for stateful layer-4 load balancing. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 1345–1358. USENIX Association, 2022.

[36] Hongxin Zeng, Lin Cui, Fung Po Tso, and Zhen Zhang. Optimizing multipath quic transmission over heterogeneous paths. *Computer Networks*, 215:109198, 2022.

[37] Jiaxing Zhang, Furong Yang, Ting Liu, Qinghua Wu, Wu Zhao, Yuanbo Zhang, Wentao Chen, Yanmei Liu, Hongyu Guo, Yunfei Ma, et al. {TECC}: Towards efficient {QUIC} tunneling via collaborative trans-

mission control. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 253–266, 2024.

[38] Menghao Zhang, Guanyu Li, Shicheng Wang, Chang Liu, Ang Chen, Hongxin Hu, Guofei Gu, Qianqian Li, Mingwei Xu, and Jianping Wu. Poseidon: Mitigating volumetric ddos attacks with programmable switches. In *the 27th Network and Distributed System Security Symposium (NDSS 2020)*, 2020.

[39] Zhilong Zheng, Yunfei Ma, Yanmei Liu, Furong Yang, Zhenyu Li, Yuanbo Zhang, Jiuhai Zhang, Wei Shi, Wentao Chen, Ding Li, et al. Xlink: Qoe-driven multi-path quic transport in large-scale video services. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, pages 418–432, 2021.

[40] Yu Zhou, Chen Sun, Hongqiang Harry Liu, Rui Miao, Shi Bai, Bo Li, Zhilong Zheng, Lingjun Zhu, Zhen Shen, Yongqing Xi, Pengcheng Zhang, Dennis Cai, Ming Zhang, and Mingwei Xu. Flow event telemetry on programmable data plane. In *Proc. of ACM SIGCOMM*, 2020.