SRAP: Sender-side Receiver-Aware Port selection for High-Speed Multi-Flow TCP

Shingo Hattori
shattori@hpcs.cs.tsukuba.ac.jp
University of Tsukuba
Tsukuba, Ibaraki, Japan

Osamu Tatebe (Advisor)
tatebe@cs.tsukuba.ac.jp
University of Tsukuba
Tsukuba, Ibaraki, Japan

Abstract

Achieving 400 Gbps requires aggregating multiple flows across cores rather than pushing single-flow limits. However, with 16×25 Gbps TCP flows, random ephemeral ports cause receiver-side packet steering (RSS) to concentrate flows on single CPU cores, degrading throughput from 25 Gbps to below 5 Gbps.

Unlike receiver-side solutions that suffer from cache misses and state migration overhead, SRAP transparently controls source ports at senders, ensuring collision-free mapping without runtime remapping costs or application modification.

With 16×25 Gbps flows, random assignment achieves optimal distribution with probability 1.04%, causing throughput to vary from 44.8 to 395 Gbps, while our approach consistently delivers 23.3-25 Gbps per flow with guaranteed 1:1 flow-to-core mapping.

CCS Concepts

• **Networks** → Network experimentation; Transport protocols; Network algorithms.

Keywords

traffic engineering, pacing, 4-tuple, flow collision, RSS, multi-flow

ACM Reference Format:
Shingo Hattori and Osamu Tatebe (Advisor). 2025. SRAP: Sender-side Receiver-Aware Port selection for High-Speed Multi-Flow TCP. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX).* ACM, New York, NY, USA, 2 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Packet pacing prevents microbursts and improves TCP performance in high-bandwidth scenarios [2, 4, 6]. Single-flow throughput is inherently limited by sequential processing constraints and CPU clock speeds. Rather than pushing against these fundamental limits, aggregating multiple flows across multiple cores provides a more scalable path to high throughput. Thus, achieving 400 Gbps requires distributing the load across multiple paced flows (e.g., 16×25 Gbps), leveraging the parallel processing capabilities of modern multi-core systems.

However, when Receive Side Scaling (RSS) maps multiple paced flows to the same CPU core, maintaining target-rate pacing becomes impossible. While RSS++ [1] attempts dynamic table adjustment at receivers, this incurs cache misses, packet reordering, and state migration overhead at 400 Gbps. This paper proposes SRAP, a transmit-side approach that prevents flow collisions through intelligent source port selection, avoiding the runtime overhead of dynamic remapping while ensuring predictable performance.

Conference acronym 'XX, Woodstock, NY
2025. ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

2 Problem Analysis

RSS distributes packets across CPU cores by hashing the 4-tuple (source/destination IPs and ports) to index an indirection table mapping to receive queues [5].

Random ephemeral port assignment (typically 32768-60999) creates unpredictable flow-to-core mappings. With 16 flows and 32 cores, the probability of collision-free assignment is merely 1.04%.

When flows collide, they must share the core's processing capacity. Figure 1 shows the extreme case—all 16 flows mapping to one core, limiting each to below 5 Gbps (44.8 Gbps aggregate). With random port assignment, collision patterns vary unpredictably from ideal 1:1 mapping to such extreme concentration, making multi-flow aggregation unreliable.

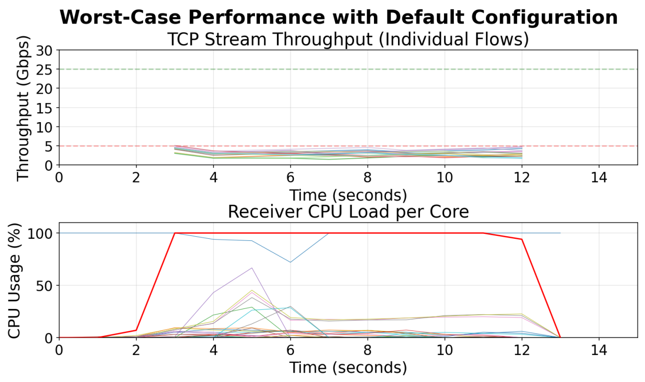

Figure 1: Random port assignment causing severe flow collisions: flows drop below 5 Gbps while one CPU reaches 100% utilization

While sender-side CPU allocation is controllable, receiver-side RSS assignment is opaque to senders. This work enables transmit-side intervention by selecting source ports to achieve predictable flow-to-core mappings.

3 Proposed Approach

SRAP controls receiver-side flow distribution by selecting source ports that hash to different CPU cores through three coordinated components (Figure 2).

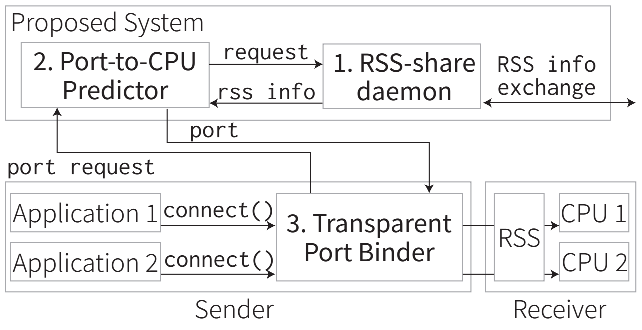

Figure 2: Proposed system architecture with three coordinated components for collision-free source port selection

(1) RSS Information Exchange: Exchanges RSS configurations (indirection tables, hash functions, IRQ mappings) with receivers for accurate port-to-core prediction.

(2) Port-to-CPU Predictor: Computes RSS hashes for potential source ports and returns collision-free ports via round-robin allocation.

(3) Transparent Port Binder: Intercepts `connect()` calls transparently, requests collision-free ports, and calls `bind()` to control the RSS 4-tuple without application modification.

By controlling source ports, SRAP prevents receiver-side flow collisions, achieving predictable multi-flow pacing without modifying applications or receivers.

4 Evaluation and Results

4.1 Experimental Setup

We evaluate between two Dell PowerEdge R6615 servers with AMD EPYC 9354P (32 cores, 3.25 GHz) and ConnectX-7 400GbE NICs running Linux 6.8.0. We use 16 parallel TCP CUBIC flows at 25 Gbps each (400 Gbps aggregate) with Linux Fair Queuing pacing [3], iperf3 with zero-copy optimizations, and 100MB TCP windows.

4.2 Evaluation

Figure 3 demonstrates SRAP's effectiveness. With our approach, all 16 flows maintain stable 23-25 Gbps throughput (top), while CPU load is evenly distributed across cores at approximately 50-65% (bottom). This contrasts sharply with the random assignment case shown in Figure 1, where flows dropped below 5 Gbps and CPUs reached 100% utilization due to collisions.

SRAP achieves perfect 1:1 flow-to-core mapping, eliminating the 99% collision probability inherent in random port assignment. Aggregate throughput consistently reaches 380-395 Gbps across all runs, compared to the 44.8-395 Gbps variance observed without control.

5 Conclusion

We demonstrated that random ephemeral port assignment causes severe RSS flow collisions, with only 1.04% probability of achieving optimal distribution for 16 flows on 32 cores, resulting in throughput varying from 44.8 to 395 Gbps.

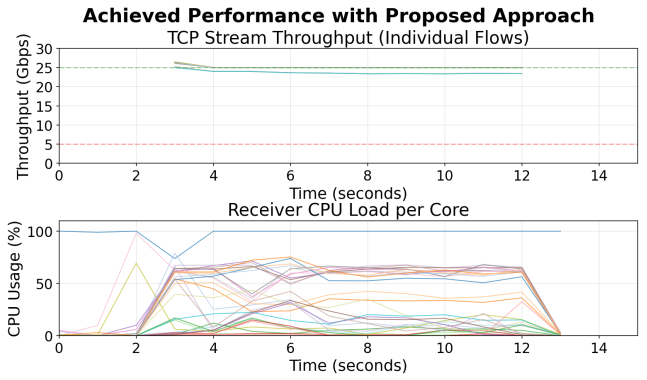

Figure 3: Achieved performance with our approach: stable 23-25 Gbps per flow with evenly distributed CPU load (50-65%)

SRAP (Sender-side Receiver-Aware Port selection) transparently controls source ports, achieving consistent 1:1 mapping and 380-395 Gbps throughput without modifying applications or receivers.

This demonstrates that source port engineering transforms unpredictable multi-flow performance into reliable high-throughput communication essential for 400G+ environments.

Acknowledgment

This work is based on results obtained from the project, "Research and Development Project of the Enhanced infrastructures for Post-5G Information and Communication Systems" (JPNP20017), commissioned by the New Energy and Industrial Technology Development Organization (NEDO), and was supported by JSPS KAKENHI Grant Number JP22H00509, JST CREST JPMJCR24R4, the Multidisciplinary Cooperative Research Program in CCS, University of Tsukuba, and Fujitsu Limited.

References

[1] Tom Barbette, Georgios P. Katsikas, Gerald Q. Maguire, and Dejan Kostić. 2019. RSS++: load and state-aware receive side scaling. In *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies* (Orlando, Florida) *(CoNEXT '19)*. Association for Computing Machinery, New York, NY, USA, 318–333. doi:10.1145/3359989.3365412

[2] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2017. BBR: congestion-based congestion control. *Commun. ACM* 60, 2 (Jan. 2017), 58–66. doi:10.1145/3009824

[3] Jake Edge. 2013. TSO sizing and the FQ scheduler. *LWN.net* (August 2013). https://lwn.net/Articles/564978/

[4] Nathan Hanford, Brian Tierney, and Dipak Ghosal. 2015. Optimizing data transfer nodes using packet pacing. In *Proceedings of the Second Workshop on Innovating the Network for Data-Intensive Science* (Austin, Texas) *(INDIS '15)*. Association for Computing Machinery, New York, NY, USA, Article 4, 8 pages. doi:10.1145/2830318.2830322

[5] Tom Herbert and Willem de Bruijn. 2015. Scaling in the Linux Networking Stack. https://www.kernel.org/doc/Documentation/networking/scaling.txt. Linux Kernel Documentation.

[6] Marcos Schwarz, Brian Tierney, Kiran Vasu, Eli Dart, Christian Esteve Rothenberg, Jeronimo Bezerra, and Italo Valcy. 2025. Recent Linux Improvements that Impact TCP Throughput: Insights from R&E Networks. In *Proceedings of the SC '24 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis* (Atlanta, GA, USA) *(SC-W '24)*. IEEE Press, 775–784. doi:10.1109/SCW63240.2024.00111

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009