Simulating MPI Collectives on Tofino Smart Switches in SimGrid

Ahmad Moh'd Saleh A. Belbeisi
ahmad.belbeisi@tum.de
Technical University of Munich
Munich, Germany

Majid Salimi Beni
majid.salimibeni@tuwien.ac.at
Vienna University of Technology
Vienna, Austria

Thomas Erbesdobler
erbesdob@cit.tum.de
Technical University of Munich
Munich, Germany

Ehab Saleh
ehab.saleh@lrz.de
Leibniz Supercomputing Centre
Munich, Germany

Matthew Tovey
Matthew.Tovey@lrz.de
Leibniz Supercomputing Centre
Munich, Germany

Amir Raoofy
amir.raoofy@lrz.de
Leibniz Supercomputing Centre
Munich, Germany

Josef Weidendorfer
Josef.Weidendorfer@tu-dresden.de
Dresden University of Technology
Dresden, Germany

Abstract

Programmable smart switches enable In-Network Computing, e.g., to accelerate HPC workloads by offloading collective operations from host CPUs. However, evaluating the benefits of these devices remains challenging due to the cost and complexity of deployment on real hardware. In this paper, we address this limitation by simulating Intel Tofino smart switches using the SimGrid framework. We take advantage of the components of the S4U and SMPI modules and introduce a new network component that represents smart switches that reproduce the latency and computational capabilities of the Tofino architecture. We validate this model on a physical testbed and present a performance evaluation of MPI collective operations offloading. Although we focus on simulating Tofino-class switches, our approach can be adapted to other smart switch architectures. Our preliminary results indicate that small-scale simulations achieve latency comparable to the real hardware when offloading `MPI_Allreduce`. This study lays the groundwork for future assessments of Tofino smart switches at scale.

Keywords

MPI, Smart Switches, Tofino, In-Network-Computing, SimGrid

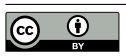
ACM Reference Format:
Ahmad Moh'd Saleh A. Belbeisi, Majid Salimi Beni, Thomas Erbesdobler, Ehab Saleh, Matthew Tovey, Amir Raoofy, and Josef Weidendorfer. 2026. Simulating MPI Collectives on Tofino Smart Switches in SimGrid. In *Proceedings of the 23rd ACM International Conference on Computing Frontiers (CF '26), May 19–21, 2026, Catania, Italy.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3801487.3801809

1 Introduction

Smart network switches are programmable devices that allow for flexible and reprogrammable routing and packet processing tasks. In Cloud and IoT, smart switches are widely deployed to enforce multi-tenancy and security policies. Providers leverage these programmable devices to dynamically isolate customer traffic, perform packet inspection, and manage virtualization resources at the edge [8]. HPC systems have historically relied on passive network fabrics. However, this paradigm is shifting; smart switches are now being used in HPC to enable In-Network Computing for tasks such as offloading collective operations, e.g., performing reduction operations. This paradigm offloads some of the reduction tasks previously handled by the nodes to the switch, accelerating inter-node communication. This overall enables more transparent and open protocols to target high performance interconnects in HPC.

Research has explored the benefits of using smart switches in HPC, showing their ability to accelerate distributed AI workloads [3, 7, 11] or offloading collective communication operations [4–6]. However, a major challenge in validating the usefulness of smart switches, in particular, Intel Tofino smart switches, is the lack of large-scale physical deployments. These switches are commonly available for smaller testbeds, and building a full-scale supercomputer with them is currently expensive and complex.

In this paper, we leverage the SimGrid simulator [2] to simulate an Intel Tofino smart switch that is dynamically programmed to accelerate MPI collective operations. We are targeting collective operations since there is a traditional interest in HPC to offload these collectives to the network, and there are existing prototypes in the literature to offload collectives to Tofino switches. We combine both the S4U and SMPI modules of SimGrid and introduce a new network element: a smart switch. In this particular case, the simulated switch performs the computations required for reduction operations and simulates the latency of real hardware during packet processing. The contributions of this paper are: ① The implementation of a smart switch model in SimGrid, combining S4U and SMPI layers; ② The simulation of a Tofino switch on a small-scale testbed; ③ A performance evaluation of MPI collective offloading on the switch, comparing the results of the physical testbed with the simulated environment.



CF '26, Catania, Italy

ACM ISBN 979-8-4007-2568-5/2026/05
https://doi.org/10.1145/3801487.3801809

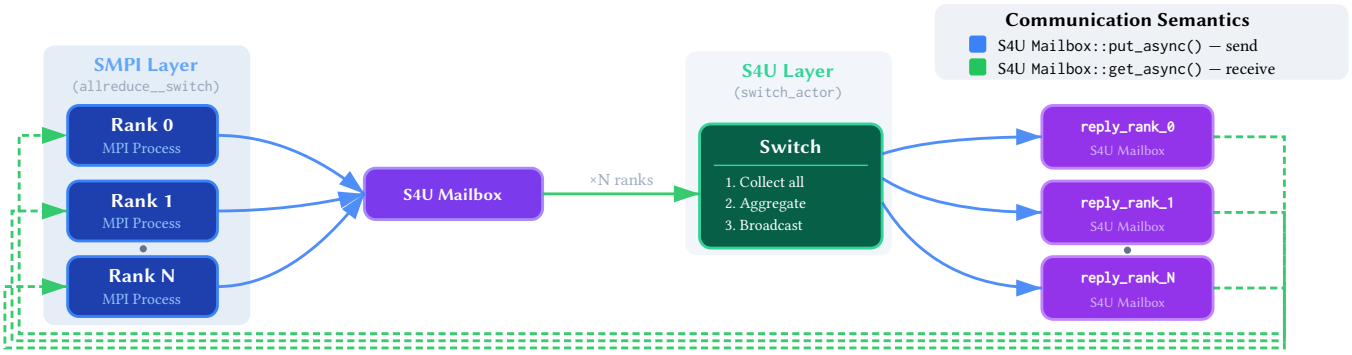


Figure 1: The simulated switch-offloaded Allreduce operation using our hybrid SMPI+S4U model.

2 Background

Tofino smart switch . Smart switches extend conventional network devices by integrating programmable data planes that allow on-the-fly custom packet processing in network. The Intel Tofino smart switch provides a multi-stage match–action pipeline implemented in a high-throughput ASIC. Its data plane is programmed using P4 [1], a domain-specific programming language that specifies packet parsing, header manipulation, and table-based forwarding behavior, and is compiled to the switch's hardware pipeline. This architecture enables the offloading of network functions such as telemetry, load balancing, and communication-aware optimizations directly into the network fabric.

SimGrid Simulator . SimGrid is a simulation framework for distributed systems. Its architecture comprises three layers: SURF, the simulation kernel that provides models for CPU, network, and disk resources; SIMIX, the kernel layer that manages process synchronization and scheduling; and user-facing APIs, including S4U and SMPI. **S4U** (SimGrid for you) is the primary C++ API for describing distributed algorithms. Applications are composed of *actors* that execute user-provided functions and interact through *activities* such as computation (`Exec`), communication (`Comm`), and I/O. Actors communicate through *mailboxes*: a sender specifies a destination mailbox, and a receiver waits on that mailbox. Network and timing information (bandwidth consumption, latency, and contention) is computed by SURF based on the platform topology. **SMPI** enables execution of unmodified MPI applications on simulated platforms. It intercepts MPI calls and replaces communication with simulated message passing. Internally, SMPI is built on top of S4U, meaning SMPI ranks are implemented as S4U actors. This architectural choice is critical for our work: SMPI and S4U share the same underlying SURF network model. An SMPI rank sending a message and an S4U actor sending to a mailbox both traverse the same bandwidth and latency simulation.

3 Related Work

Several studies have shown the efficacy of offloading tasks to programmable network devices, particularly Smart Switches. Hoefler et al. [7] explore the opportunities for in-network collective operations to accelerate AI workloads. Cui et al. [3] present *NetFC*, a table-lookup method to achieve on-the-fly in-network floating-point arithmetic operations for Tofino switches. Similarly, Sapio et al. [11] presented *SwitchML*, a programmable switch co-designed with end-host protocols that performs in-network aggregation of deep learning model updates to reduce data exchange. Erbesdobler et al. [4] introduce *mpitofino*, a prototype integrated into Open MPI that utilizes P4-enabled Tofino switches to offload collective reductions over RoCEv2. Recent research has also explored approaches leveraging SmartNICs (DPUs) for task offloading. *ODOS-MPI* [12] is a framework that leverages DPUs to offload both computation and MPI communication kernels, enabling efficient overlap of computation and data movement. Focusing on one-sided communication, Michalowicz et al. [9] use BlueField DPUs to offload MPI and OpenSHMEM primitives, showing that delegating communication to the SmartNIC results in performance improvements for SPMM workloads. Graham et al. [5] present DPU-offloaded algorithms for `MPI_Ialltoallv` and `MPI_Allgatherv`, showing performance improvements for real-world workloads by mitigating load imbalance.

Related work has shown the benefits of offloading collective operations to smart switches; however, the practical benefits of employing smart switches in HPC have not been studied. This highlights the need for a simulation study of these switches to enable designing future systems based on them, e.g., enabling larger-scale simulations before physical deployment.

4 Modeling Smart Switches in SimGrid

This section details our strategy for modeling in-network compute, in particular, MPI collectives on Tofino switches.

Our simulated framework must meet these requirements: abstraction fidelity to model the switch as an external device that sees packets; performance accuracy to simulate realistic bandwidth, latency, and delays; selective offloading to use the switch for supported collectives while defaulting to host algorithms otherwise; application transparency to run existing MPI apps unchanged; and validation tractability to enable tracing for correctness and routing.

SimGrid's SMPI framework provides validated models for host-based collective operations but assumes all computation occurs on MPI ranks. In-network computing introduces computation at an intermediate layer: below the MPI runtime, which manages communicators and collective semantics, and above packet-level simulation. We exploit SimGrid's flow-based network model, which computes transfer times analytically from bandwidth and latency without simulating individual packets, to construct a hybrid simulation where SMPI ranks coexist with an S4U switch actor performing message-granularity aggregation.

```
<platform version="4.1">
  <!-- Smart Switch -->
  <host id="switch0" speed="100Gf">
  <prop id="processing_latency" value="3e-6"/>
   <prop id="max_ports" value="16"/>
  </host>
</platform>
```

Figure 2: An example of a smart switch in SimGrid allowing nodes to connect to it via 100 Gb/s links.

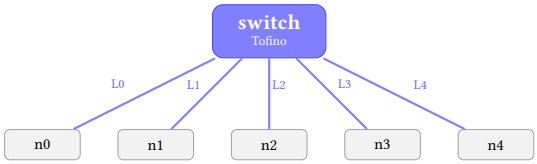


Figure 3: Our experimental setup's topological view with nodes connected to the switch via dedicated links.

4.1 Hybrid SMPI/S4U Architecture

We model the switch as an S4U actor exchanging messages with SMPI ranks, which mirrors how physical MPI processes and switches interact only at the network level. Figure 1 illustrates the different components in the simulated model. In this figure, the switch actor (the green box) models a programmable switch performing in-network reduction and has adjustable parameters for processing latency. The switch is practically responsible for three main tasks during an Allreduce operation:

Collection: Pre-post N asynchronous receives on the shared ingress mailbox, where N is the communicator size established at initialization. Wait for all receives to complete in parallel, enabling concurrent data transfer from all ranks.

Reduction: Perform the reduction operation (element-wise sum, in our case) and model the switch using a single end-to-end latency from data ingress to data egress.

Distribution: Transmit results to each rank's dedicated reply mailbox via asynchronous sends, with flow-based network timing determining transfer completion.

Our simulation operates at message granularity using SimGrid's flow-based network model. Transfer time is modeled from message size, link bandwidth, and latency, with automatic handling of contention on shared links. No individual packets are simulated; each mailbox operation transfers a complete message atomically.

4.2 Simulating MPI Collectives on Tofino

We simulate a Tofino-class smart switch in SimGrid to study in-network offloading while keeping the switch logically independent from the MPI runtime. The simulation comprises four parts: a calibrated platform description, an MPI interception layer, an S4U switch actor, and an application execution mode.

Platform definition. We define the SimGrid platform in an XML file, specifying compute hosts, the switch host, and network links. For link latencies, we run a p2p benchmark on all pairs on real hardware and set the link latencies in the simulator to half the round-trip time measured in the benchmark. The switch latency is determined by running an `MPI_Allreduce` microbenchmark and calculating the difference between the link latency and the collective operation's overall latency. In reality, latency cannot be traced back to per-link components or compute resources within the switch ASIC because the architecture is pipelined. To account for this, we empirically adjusted the processing latency in our model until the simulated behavior most closely matched the real system's behavior. Figure 2 shows an example of a smart switch definition in the platform's XML file, where the parameters can be adjusted based on the type of switch.

MPI interception and offload policy. The `MPI_Allreduce` calls are intercepted in the *switch*, and we decide whether to offload or fall back. Our current implementation supports `MPI_INT` with `MPI_SUM` as data type and operation, respectively, which reflects constrained data types and operations in Tofino switches. All other combinations fall back to SimGrid's `redbcast` to ensure correctness.

Switch actor workflow. The switch is modeled as an S4U actor running on the switch host, is completely outside the MPI runtime, and is not assigned an MPI rank and communicator. Ranks send a request message (rank ID, communicator size, element count and size, reply mailbox name) and the data to a shared mailbox; the actor performs the reduction and returns the result via per-rank reply mailboxes.

Application execution. MPI applications run unmodified through SMPI dynamic loading, requiring no source code changes.

This abstraction aligns with our goal of predicting collective communication completion times rather than simulating packet dynamics. Enabling parameter sweeps over topology and timing configurations would be expensive in packet-level simulation. The trade-off is that we do not capture in-cast congestion or buffer overflows; in a single-switch star topology, these effects are minimal.

5 Experimental Evaluation

This section evaluates the simulated smart switch against a physical testbed, ensuring three validation rules. First, we verify functional correctness by ensuring that the results of `MPI_Allreduce` match in both environments. Second, we validate the switch control logic by inspecting switch-side counters that indicate whether a collective operation is handled by the supported (offloaded) datapath or by a fall back path; supported configurations must increment the corresponding offload counters, while unsupported configurations must leave them unchanged. Third, we validate the modeled topology using SimGrid's route API to confirm that all the traffic is routed through the switch, ensuring that both the simulator and the testbed follow the same communication structure.

5.1 Experimental Setup

Our evaluation platform consists of a smart switch connected to 5 nodes in a star topology, as shown in Figure 3. It consists of a STORDIS BF6064X-T Ethernet switch based on an Intel Tofino ASIC with a switching capacity of 6.4 Tb/s. The switch control plane runs on an Intel Xeon D-1548 CPU at 2.00 GHz. We use five compute nodes, each equipped with 2× Intel Xeon Platinum 8360 CPUs. Each node connects to the switch via a single 100 Gb/s 100GBase-CR4 link and uses a ConnectX-6 Dx NIC (as part of a BlueField-2 DPU configured in NIC mode). *mpitofino* library [4] is used for task offloading to the switch, and OSU microbenchmarks [10] are used in the experiments.

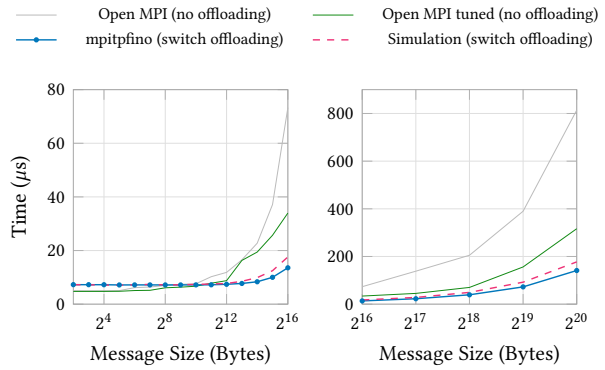


Figure 4: `MPI_Allreduce` latency comparison on different message sizes: Simulation vs *mpitofino* vs Open MPI (Basic and Tuned) on 4×1 processes.

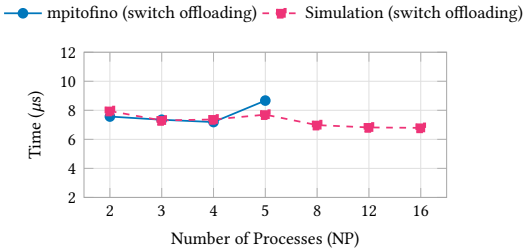


Figure 5: `MPI_Allreduce` latency for 1 KiB of *mpitofino* vs simulation, with scaling simulation to up to 16×1 processes.

5.2 Experimental Results

Figure 4 shows the performance of Open MPI, in both its basic and built-in MCA tuned collective algorithms, *mpitofino*, which supports switch offloading, and our simulation of switch offloading for a range of message sizes. Here, we clearly see the benefits of offloading reductions to the switch, where *mpitofino* is 2.2× faster than tuned Open MPI. Most importantly, our simulated switch can accurately replicate the latency of offloading `MPI_Allreduce`. Across all tested message sizes, the average deviation between real-world switch offloading (*mpitofino*) and the simulation for this collective operation is about 10% (~4.3 μs).

To evaluate scalability in the star topology, we increased the number of nodes to 8, 12, and 16 in the simulation study. Figure 5 shows the results where the simulation yields similar latency when performing `MPI_Allreduce` and the time difference between them is smaller than 1 μs. This figure also shows that increasing the number of nodes connected to a Tofino switch does not increase the latency of the Allreduce operation for small-to-medium message sizes. This is mainly because the switch assists with reduction operations, and since the links are not fully saturated for smaller message sizes, there is no slowdown because of link congestion.

6 Conclusion

We presented a novel simulation model within the SimGrid framework. By coupling the S4U and SMPI modules, we reproduced the computational capabilities and communication latency of smart switches handling MPI collective offloading. We validated this model against a physical testbed equipped with Intel Tofino switches, demonstrating that our simulation accurately captures the performance characteristics of real hardware. While this study focused on characterizing the Intel Tofino architecture, the proposed simulation methodology is generic and can be adapted to simulate other programmable switches. Current limitations include SimGrid's flow-level CM02 network model, which does not capture packet-level pipelining, and simplified intra-node communication modeling when multiple MPI processes share a node.

Future work will extend the current implementation to evaluate larger, complex topologies. Specifically, we plan to refine the mailbox-based model to improve accuracy; improve intra-node modeling for multiple processes; test multi-switch topologies (fat-tree, dragonfly); and implement additional collective operations.

Acknowledgments

This paper is supported by funding from the ScalNEXT project (number 16ME0687) by the BMFTR. Performance results were obtained on BEAST systems at LRZ.

References

[1] Pat Bosshart, Dan Daly, Glen Gibb, et al. 2014. P4: programming protocol-independent packet processors. *Comput. Commun. Rev.* 44, 3 (2014), 87–95. doi:10.1145/2656877.2656890

[2] Henri Casanova, Arnaud Giersch, Arnaud Legrand, et al. 2014. Versatile, scalable, and accurate simulation of distributed applications and platforms. *J. Parallel Distributed Comput.* 74, 10 (2014), 2899–2917. doi:10.1016/J.JPDC.2014.06.008

[3] Penglai Cui, Heng Pan, Zhenyu Li, et al. 2022. Enabling In-Network Floating-Point Arithmetic for Efficient Computation Offloading. *IEEE Trans. Parallel Distributed Syst.* 33, 12 (2022), 4918–4934. doi:10.1109/TPDS.2022.3208425

[4] Thomas Erbesdobler, Amir Raoofy, Ehab Saleh, and Josef Weidendorfer. 2025. MPI Collectives with Programmable Smart Switches. In *Proceedings of the SC '25 Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 468–478. doi:10.1145/3731599.3767391

[5] Richard L. Graham, George Bosilca, Yong Qin, et al. 2024. Optimizing Application Performance with BlueField: Accelerating Large-Message Blocking and Nonblocking Collective Operations. In *ISC High Performance*. 1–12. doi:10.23919/ISC.2024.10528935

[6] Richard L. Graham, Devendar Bureddy, Pak Lui, et al. 2016. Scalable Hierarchical Aggregation Protocol (SHArP): A Hardware Architecture for Efficient Data Reduction. In *1st International Workshop on Communication Optimizations in HPC*. 1–10. doi:10.1109/COMHPC.2016.006

[7] Torsten Hoefler, Mikhail Khalilov, Josiah Clark, Surendra Anubolu, Mohan Kalkunte, Karen Schramm, Eric Spada, Duncan Roweth, Keith Underwood, Adrian Caulfield, Abdul Kabbani, and Amirreza Rastegari. 2026. In-Network Collective Operations: Game Changer or Challenge for AI Workloads? *Computer* 59, 1 (2026), 24–33. doi:10.1109/MC.2025.3616048

[8] Sajy Khashab, Alon Rashelbach, and Mark Silberstein. 2024. Multitenant In-Network Acceleration with SwitchVM. In *21st USENIX Symposium on Networked Systems Design and Implementation, NSDI*. 691–708. https://www.usenix.org/conference/nsdi24/presentation/khashab

[9] Benjamin Michalowicz, Kaushik Kandadi Suresh, Hari Subramoni, et al. 2024. Effective and Efficient Offloading Designs for One-Sided Communication to SmartNICs. In *31st IEEE International Conference on High Performance Computing, Data, and Analytics (HiPC)*. IEEE, 23–33. doi:10.1109/HIPC62374.2024.00012

[10] Network-Based Computing Laboratory. [n. d.]. OSU Micro-Benchmarks. https://mvapich.cse.ohio-state.edu/benchmarks/. The Ohio State University.

[11] Amedeo Sapio, Marco Canini, Chen-Yu Ho, et al. 2021. Scaling Distributed Machine Learning with In-Network Aggregation. In *18th USENIX Symposium on Networked Systems Design and Implementation, NSDI*. 785–808. https://www.usenix.org/conference/nsdi21/presentation/sapio

[12] Muhammad Usman, Mariano Benito, Sergio Iserte, and Antonio J. Peña. 2025. ODOS-MPI: HPC-Friendly SmartNIC Offloading of Computation/Communication Kernels. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. 1006–1027. doi:10.1145/3712285.3759808