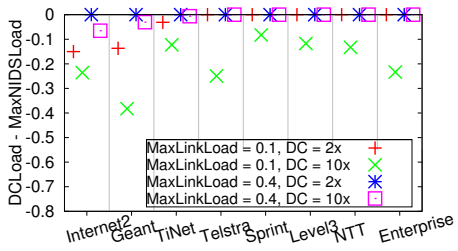

Figure 12: Comparing the compute load on the datacenter vs. maximum load on interior NIDS nodes.

nodes gets a $\frac{1}{|\mathcal{N}|}$ share of the $10\times$ additional resources. The fact that we can consider these alternative designs within our framework further confirms the *generality* of our approach.

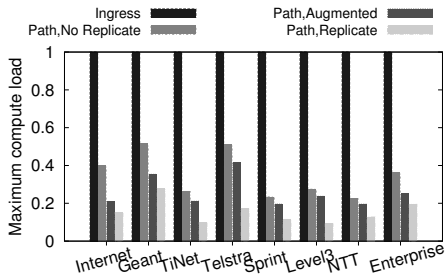

Figure 13: Maximum compute load across topologies with different NIDS architectures.

Recall that the current deployments of *Ingress*-only have a maximum compute load of one by construction. The result shows that *Path, Replicate* has the best overall performance; it can reduce the maximum compute load by $10\times$ compared to today's deployments and up to $3\times$ compared to the proposed on-path distribution schemes.

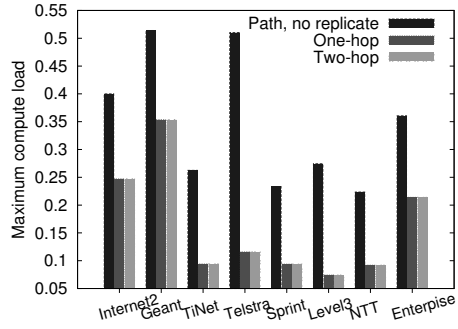

Figure 14: Local one- and two-hop replication.

Local offload: The above results consider a setup where the network administrator has added a new datacenter. Alternatively, they can use the existing NIDS infrastructure with *local* replication strategies. Specifically, we consider the mirror sets (M_js) consisting of 1-hop or 2-hop neighbors in addition to the existing on-path distribution. Figure 14 compares the maximum compute load vs. a pure on-path distribution again setting $MaxLinkLoad = 0.4$. Across all topologies, allowing replication within a one-hop radius provides up to $5\times$ reduction in the maximum load. We also see that going to two hops does not add significant value beyond one-hop offload. This suggests a replication-enhanced NIDS architecture can offer significant benefits even without needing to augment the network with additional compute resources.

Performance under traffic variability: The results so far consider a static traffic matrix. Next, we evaluate the effect of traffic variability. To obtain realistic temporal variability patterns, we use traffic matrices for Internet2 [14]. From this, we compute empirical CDFs of how each element in a traffic matrix varies (e.g., probability that the volume is between $0.6\times$ and $0.8\times$ the mean). Then, using these empirical distributions we generate 100 time-varying traffic matrices using the gravity model for the mean volume.

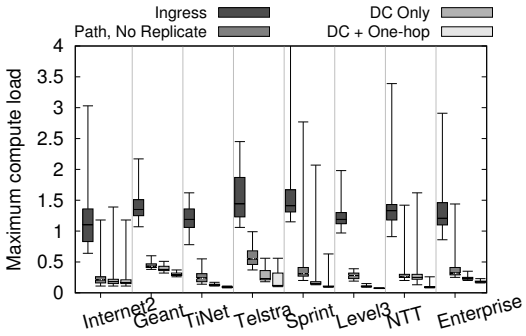

Figure 15: Comparison between NIDS architectures in the presence of traffic variability.

Figure 15 summarizes the distribution of the peak load across these 100 runs using a box-and-whiskers plot showing the minimum, 25th %ile, median, 75th %ile, and the maximum observed load. We consider four NIDS architectures: *Ingress*; *Path, No replicate*; *Path, replicate* with a datacenter node $10\times$ capacity (labeled `DC Only`); and *Path, replicate* with the flexibility to offload responsibilities to either a datacenter and within a 1-hop radius (labeled `DC + One-hop`). We find that the replication-enabled NIDS architectures outperform the non-replication strategies significantly, with the median values roughly mirroring our earlier results. The worst-case performance of the no-replication architectures can be quite poor, e.g., much larger than 1. (Ideally, we want the maximum compute load to be less than 1.) We also analyzed how the augmentation strategy from Figure 13 performs; the worst-case load with the *Path, Augmented* case is $4\times$ more than the replication enabled architecture (not shown).

8.3 Replication with routing asymmetry

In this section, we evaluate how replication is effective for scenarios where the forward and reverse flows may not traverse the same route as we saw in Section 2.

We emulate routing asymmetry as follows. For each ingress-egress pair, we assume the forward traffic traverses the expected shortest path from the ingress to the egress; i.e., P_c^{fwd} is the shortest-path route. However, we set the reverse path P_c^{rev} such that the expected *overlap* (over all ingress-egress pairs) between the forward and reverse paths reaches a target overlap ratio θ. Here, we measure the overlap between two paths P_1 and P_2 in terms of the Jaccard similarity index: $\frac{P_1 \cap P_2}{P_1 \cup P_2}$, which is maximum (= 1) when they are

identical and lowest (= 0) if there is no overlap. For each end-to-end path, we precompute its overlap metric with every other path. Then, given a value of θ' (drawn from a Gaussian distribution with mean = θ and standard deviation = $\frac{\theta}{5}$), we find a path from this precomputed set that is *closest* to this target value.[8] For each target θ, we generate 50 random configurations. For each configuration, we run the extended formulation from Section 5 for the *Ingress*-only architecture, the *Path, no replicate* architecture, and our proposed framework with a datacenter. We report the *median* across the 50 runs for two metrics: the detection *miss rate* — the total fraction of traffic that could not be analyzed effectively by any NIDS node — and the compute load as in the previous evaluations.

Figure 16 shows the median miss rate as a function of the overlap factor for the different configurations. We see that the miss rate with an *Ingress*-only setup is greater than 85% even for high values of the overlap metric. The $MaxLoad$ curve in Figure 17 is interesting because *Ingress* is lower than the other configurations. The reason is that there is little useful work being done here — It ignores more than 90% of the traffic! Another curious feature is that $MaxLoad$ for the replication architecture first increases and then decreases. In this setup with low overlap, the datacenter is the most loaded node. At low θ, however, the $MaxLinkLoad$ constraint limits the amount of traffic that can be offloaded and thus the datacenter load is low.

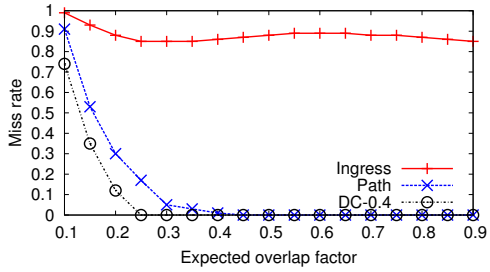

Figure 16: Detection miss rate vs. degree of overlap

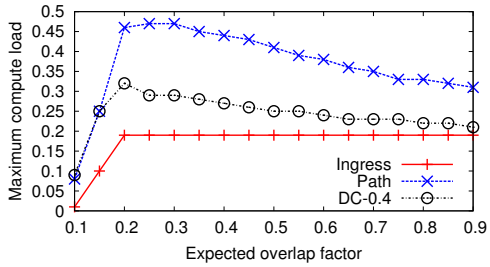

Figure 17: Maximum load vs. degree of overlap

8.4 NIDS with aggregation

In this section, we highlight the benefits of aggregation using the framework from Section 6. As discussed earlier, we focus on `Scan` detection.

[8]The exact details of how these paths are chosen or the distribution of the θ values are not the key focus of this evaluation. We just need some mechanism to generate paths with a target overlap ratio.

Figure 18 shows how varying β trades off the communication cost ($CommCost$) and compute cost ($LoadCost$) in the resulting solution, for the different topologies. Because different topologies differ in size and structure, we normalize the x- and y-axes using the maximum observed $LoadCost$ and $CommCost$ respectively over the range of β for each topology. As such, the point closest to the origin can be viewed as the best choice of β for the corresponding topology. This figure shows that for many topologies, there are choices of β that yield relatively low $CommCost$ and $LoadCost$ simultaneously, e.g., both being less than 40% of their maximums.

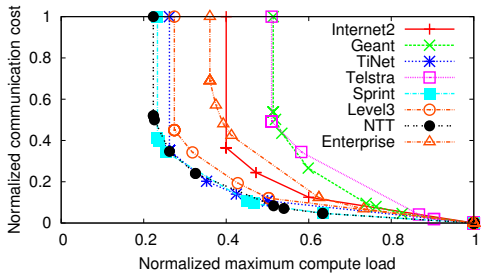

Figure 18: Tradeoff between the compute load and communication cost with aggregation as we vary β

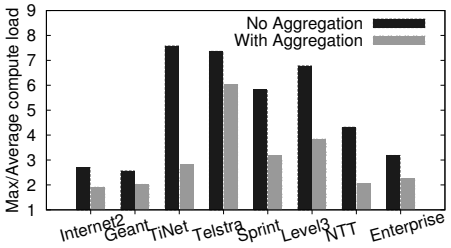

Figure 19: Ratio between maximum and average compute load with and without aggregation.

To illustrate the load balancing benefits of aggregation, Figure 19 shows the ratio of the compute load of the most loaded node to the average compute load. Here, for each topology, we pick the value of β that yields the point closest to the origin in Figure 18. A higher number represents a larger variance or imbalance in the load. Figure 19 compares this ratio to the same ratio when no aggregation is used. As we can see, aggregation reduces the load imbalance substantially (up to $2.7\times$) for many topologies.

8.5 Summary of key results

Our main results are:

- The optimization step and shim impose low overhead.
- Administrators need not worry about optimal choice of data center location, capacity, or the maximum link load. Our approach provides benefits over a range of practical and intuitive choices.
- Replication reduced the maximum compute load by up to $10\times$ when we add a NIDS cluster or up to $5\times$ with one-hop offload.

- In the presence of traffic dynamics, replication provided up to an order of magnitude reduction in maximum load.
- Replication reduced the detection miss rate from 90% to zero in the presence of partially overlapping routes.
- Aggregation reduced the load imbalance by up to $2.7\times$.

9. DISCUSSION

Consistent configurations: One concern with distribution is ensuring consistency when configurations are recomputed. We could use standard techniques from the distributed systems literature (e.g., two-phase commit [21]). We can also use simpler domain-specific solutions; e.g., whenever new configurations are pushed out, the NIDS nodes continue to honor both the previous and new configurations during the transient period. This may potentially duplicate some work, but ensures correctness of operation.

Shim for higher line-rates: Our current shim implementation imposes close to zero overhead for a single-threaded NIDS running on a single core machine for traffic up to 1 Gbps. We plan to extend our implementation using recent advances in packet capture [25, 26].

Robustness to dynamics: A sudden, significant shift in traffic patterns (adversarial or otherwise) could render the current distribution strategies ineffective. One approach to counter this is to allow for some "slack" (e.g., using the 80-th percentile values instead of the mean) in the input traffic matrices to tolerate such sudden bursts.

Extending to NIPS and active monitoring: Our approach can be generally applied to any *passive* traffic monitoring system without affecting the forwarding paths or latency of traffic. Our framework can also be extended to the case of intrusion prevention systems (NIPS), though unlike NIDS, NIPS are on the critical forwarding path which raises two additional issues that we need to handle. These arise from the fact that we are not actually replicating traffic in this case; rather, we are *rerouting* it. First, we can no longer treat the BG_l as a constant in the formulation. Second, we need to ensure that the latency penalty for legitimate traffic due to rerouting is low.

Combining aggregation and replication: As future work we plan to explore if a unified formulation that combines both opportunities offers further improvements. For example, we might be able to use replication to reduce the communication cost of aggregation. One challenge is that the analyses benefiting from aggregation may need to split the traffic at a different granularity (e.g., per source) vs. those exploiting replication (e.g., stateful signature matching on a per-session basis). Thus, we need a more careful shim design to avoid duplicating the effort in packet capture across different nodes in order to combine these ideas.

10. RELATED WORK

Scaling NIDS hardware: NIDS involve computationally intensive tasks (e.g., string and regular-expression matching). There are many proposals for better algorithms for such tasks (e.g., [33]), using specialized hardware such as TCAMs (e.g., [20, 42]), FPGAs (e.g., [17]), or GPUs (e.g., [38]). The dependence on specialized hardware increases deployment costs. To address this cost challenge, there are ongoing efforts to build scalable NIDS on commodity hardware to exploit data-level parallelism in NIDS workloads (e.g., [37,39]). These efforts focus on scaling *single-vantage-point* implementations and are thus complementary to our work. Our framework allows administrators to optimally use their existing hardware or selectively add NIDS clusters.

NIDS management: Our use of centralized optimization to assign NIDS responsibilities follows in the spirit of our prior work [29]. The approach we propose here extends our prior work in three key ways. First, we generalize on-path distribution to include replication and aggregation. Second, this previous framework cannot handle the types of split-traffic analysis with asymmetric routes as we showed in Figure 16. Third, on a practical note, this past approach requires source-level changes to the NIDS software. In contrast, our implementation allows administrators to run off-the-shelf NIDS software.

Offloading NIDS: A recent proposal makes a case for outsourcing all network processing functionality including NIDS to cloud providers [32]. While this may work for small businesses and enterprises, larger enterprises and ISPs would likely retain in-house infrastructure due to security and policy considerations. Furthermore, this proposal does not focus on computation-communication tradeoffs. Our approach can also incorporate a cloud datacenter and can offer ways to augment existing infrastructure instead of getting rid of it altogether.

Distributed NIDS: Prior work makes the case for network-wide visibility and distributed views in detecting anomalous behaviors (e.g., [16]). These focus primarily on algorithms for combining observations from multiple vantage points. Furthermore, specific attacks (e.g., DDoS attacks, stepping stones) and network scenarios (e.g., asymmetric routing as in Section 2) inherently require an aggregate view. Our focus is not on the algorithms for combining observations; rather, we build a framework for enabling such aggregated analysis.

11. CONCLUSIONS

While there are many advances in building better NIDS hardware, there is a substantial window before networks can benefit from these in practice. Our work complements existing research in scaling NIDS hardware with techniques to better utilize and augment existing NIDS deployments. To this end, we proposed a general NIDS architecture to leverage three opportunities: offloading processing to other nodes on a packet's routing path, traffic replication to off-path nodes (e.g., to NIDS clusters), and aggregation to split expensive NIDS tasks. We implemented a lightweight shim that allows networks to realize these benefits with little to no modification to existing NIDS software. Our results on many real-world topologies show that this architecture reduces the maximum compute load significantly, provides better resilience under traffic variability, and offers improved detection coverage for scenarios needing network-wide views.

Acknowledgements

We are grateful to Geoff Voelker for commenting on drafts of this paper and to Chad Spensky for initial discussions on this research. This work was supported in part by NSF grants 0831245 and 1040626, and by grant number N00014-10-1-0155 from the Office of Naval Research.

12. REFERENCES

[1] Powering virtual network services. http://embrane.com.
[2] ZScaler Cloud Security. http://www.zscaler.com.
[3] Allman, M., Kreibich, C., Paxson, V., Sommer, R., and Weaver, N. Principles for developing comprehensive network visibility. *In Proc. HOTSEC'08*, 2008.
[4] Bittwist. http://bittwist.sourceforge.net.
[5] Bob hash. http://burtleburtle.net/bob/hash/doobs.html.

[6] Cantieni, G. R., Iannaccone, G., Barakat, C., Diot, C., and Thiran, P. Reformulating the monitor placement problem: optimal network-wide sampling. *In Proc. CoNEXT '06*, 2006.

[7] Cisco blade servers. http://www.cisco.com/en/US/products/ps10280/index.html.

[8] Dreger, H., Feldmann, A., Paxson, V., and Sommer, R. Predicting the resource consumption of network intrusion detection systems. *In Proc. SIGMETRICS '08*, 2008.

[9] Feldmann, A., Greenberg, A., Lund, C., Reingold, N., Rexford, J., and True, F. Deriving traffic demands for operational IP networks: methodology and experience. *IEEE/ACM Trans. Netw.*, 9(3):265–280, June 2001.

[10] Fortz, B., Rexford, J., and Thorup, M. Traffic engineering with traditional IP routing protocols. *Communications Magazine, IEEE*, 40(10):118 – 124, Oct 2002.

[11] Gibb, G., Zeng, H., and McKeown, N. Outsourcing network functionality. In *Proc. HotSDN*, 2012.

[12] Greenberg, A., Hjalmtysson, G., Maltz, D. A., Myers, A., Rexford, J., Xie, G., Yan, H., Zhan, J., and Zhang, H. A clean slate 4D approach to network control and management. *SIGCOMM Comput. Commun. Rev.*, 35(5):41–54, Oct. 2005.

[13] Heorhiadi, V., Reiter, M. K., and Sekar, V. Balancing computation-communication tradeoffs in scaling network-wide intrusion detection systems. Technical Report TR12-001, UNC Chapel Hill, 2012.

[14] Internet2 trafficx matrices. http://www.cs.utexas.edu/~yzhang/research/AbileneTM.

[15] Kohler, E., Morris, R., Chen, B., Jannotti, J., and Kaashoek, M. F. The click modular router. *ACM Trans. Comput. Syst.*, 18(3):263–297, Aug. 2000.

[16] Lakhina, A., Crovella, M., and Diot, C. Diagnosing network-wide traffic anomalies. *In Proc. SIGCOMM '04*, 2004.

[17] Lee, J., Hwang, S. H., Park, N., Lee, S.-W., Jun, S., and Kim, Y. S. A high performance NIDS using FPGA-based regular expression matching. *In Proc. SAC '07*, 2007.

[18] M57 packet traces. https://domex.nps.edu/corp/scenarios/2009-m57/net/.

[19] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. OpenFlow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, Mar. 2008.

[20] Meiners, C. R., Patel, J., Norige, E., Torng, E., and Liu, A. X. Fast regular expression matching using small TCAMs for network intrusion detection and prevention systems. *In Proc. USENIX Security'10*, 2010.

[21] Mohan, C. and Lindsay, B. Efficient commit protocols for the tree of processes model of distributed transactions. *SIGOPS Oper. Syst. Rev.*, 19(2):40–52, Apr. 1985.

[22] Network security spending to soar in the next 5 year. http://www.v3.co.uk/v3-uk/news/1998293/network-security-spending-soar.

[23] PAPI: Performance application programming interface. http://icl.cs.utk.edu/papi/.

[24] Paxson, V. Bro: a system for detecting network intruders in real-time. *In Proc. Usenix Security'98*, 1998.

[25] Pfq homepage. http://netserv.iet.unipi.it/software/pfq/.

[26] Pf_ring. http://www.ntop.org/products/pf_ring/.

[27] Roughan, M. Simplifying the synthesis of internet traffic matrices. *SIGCOMM Comput. Commun. Rev.*, 35(5):93–96, Oct. 2005.

[28] Scapy packet manipulation toolkit. http://www.secdev.org/projects/scapy/.

[29] Sekar, V., Krishnaswamy, R., Gupta, A., and Reiter, M. K. Network-wide deployment of intrusion detection and prevention systems. *In Proc. CoNEXT '10*, 2010.

[30] Sekar, V., Ratnasamy, S., Reiter, M. K., Egi, N., and Shi, G. The middlebox manifesto: enabling innovation in middlebox deployment. *In Proc. HotNets '11*, 2011.

[31] Shaikh, A. and Greenberg, A. Ospf monitoring: architecture, design and deployment experience. *In Proc. NSDI'04*, 2004.

[32] Sherry, J., Hasan, S., Scott, C., Krishnamurthy, A., Ratnasamy, S., and Sekar, V. Making middleboxes someone else's problem: Network processing as a cloud service. *In Proc. SIGCOMM*, 2012.

[33] Smith, R., Estan, C., and Jha, S. XFA: Faster signature matching with extended automata. *In Proc. IEEE S&P'08*, 2008.

[34] Snort. http://www.snort.org.

[35] Spring, N., Mahajan, R., Wetherall, D., and Anderson, T. Measuring isp topologies with rocketfuel. *IEEE/ACM Trans. Netw.*, 12(1):2–16, Feb. 2004.

[36] Teixeira, R., Shaikh, A., Griffin, T., and Rexford, J. Dynamics of hot-potato routing in IP networks. *In Proc. SIGMETRICS '04/Performance '04*, 2004.

[37] Vallentin, M., Sommer, R., Lee, J., Leres, C., Paxson, V., and Tierney, B. The NIDS cluster: scalable, stateful network intrusion detection on commodity hardware. *In Proc. RAID'07*, 2007.

[38] Vasiliadis, G., Polychronakis, M., Antonatos, S., Markatos, E. P., and Ioannidis, S. Regular expression matching on graphics hardware for intrusion detection. *In Proc. RAID '09*, 2009.

[39] Vasiliadis, G., Polychronakis, M., and Ioannidis, S. MIDeA: a multi-parallel intrusion detection architecture. *In Proc. CCS '11*, 2011.

[40] White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., and Joglekar, A. An integrated experimental environment for distributed systems and networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):255–270, Dec. 2002.

[41] World intrusion detection and prevention markets. http://www-935.ibm.com/services/us/iss/pdf/esr_intrusion-detection-and-prevention-systems-markets.pdf.

[42] Yu, F., Lakshman, T. V., Motoyama, M. A., and Katz, R. H. SSA: a power and memory efficient scheme to multi-match packet classification. *In Proc. ANCS '05*, 2005.

[43] Zhang, Y. and Paxson, V. Detecting stepping stones. *In Proc. Usenix Security'00*, 2000.