Transparent TCP Acceleration Through Network Processing

Tilman Wolf, Shulin You, and Ramaswamy Ramaswamy

Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA 01003
{wolf,syou,rramaswa}@ecs.umass.edu

***Abstract*— Transparent TCP acceleration is an approach to increasing TCP throughput without requiring any changes in end-system TCP implementations. Network processing technology is used to intercept and transparently relay TCP connections inside the network. Due to the transparent nature of the system, incremental deployment and opportunistic acceleration is easily possible. The simulation results show that a single acceleration node can improve throughput twofold. More acceleration nodes can further increase the performance improvement, particularly on lossy links.**

I. Introduction

TCP is a commonly used transport layer protocol in the Internet. It provides reliability, flow control, and congestion control services on top of the lossy, best-effort network layer. There are a number of control mechanisms that are implemented in TCP that determine when a packet is considered lost, when to retransmit, and when to slow down the rate of transmission. Over the past decades, many components of the TCP protocol have been fine tuned to incrementally increase the overall TCP performance. In all these approaches, the network itself was considered an interconnect that simply moves data from one end-system to another. Only small changes in the network functionality were considered (e.g., RED, FRED, etc.) to support TCP performance.

In our work, we make use of recently developed network processor technology to implement TCP accelerators *inside* the network. We show that this approach can significantly improve the performance of TCP connections and it can be incrementally deployed. The significance and potential impact of these results is considerable since the deployment of such TCP accelerators improves the performance that any network user can experience.

The idea of a TCP accelerator is to use a network processor, which is typically located on router systems, to terminate TCP connections inside the network processor and then relay the data to a second connection towards the end system. It is possible to have numerous TCP accelerators in a single end-to-end connection. The performance improvement comes from reducing the delay in the feedback loop that can trigger retransmissions. It is important to note that the feedback loop is always active and retransmissions are often necessary due to the TCP behavior of continuously increasing the transmission bandwidth until packet loss occurs. As a result, even under ideal conditions (no packet loss due to link errors and no contention for bandwidth), retransmissions happen. The shorter the feedback loop is to trigger the retransmission, the higher the overall performance of the connection.

In order to implement a TCP accelerator, it is necessary to augment routers with the ability to store packets and perform TCP-compliant processing of packets. Network processors (NPs) are ideal systems to perform this task. NPs are usually implemented as embedded multiprocessor systems-on-a-chip with considerable amounts of processing power and memory and are software programmable.

In this paper, we discuss the practical issues of implementing and deploying TCP accelerators as well as present simulation results that show the potential performance improvements given practical constraints on placement and memory capacity. We show that it is feasible to implement a TCP accelerator that supports hundreds to thousands of simultaneous connections on an off-the-shelf network processor. The simulation results show that even a single acceleration node can achieve twofold throughput improvement. On lossy links and with more nodes, even higher throughput speedup is possible.

Section II discusses related work. Section III discusses the operational and implementation details of an acceleration node. Section IV discusses deployment issues and Section V presents simulation results.

II. Related Work

There has been a large body of work on improving TCP throughput, which has mainly focused on developing different flavors of TCP (e.g., Vegas vs. Reno) on the end-system. We are addressing acceleration techniques inside the network.

A. TCP Acceleration Techniques

The performance of an end-to-end transport protocol degrades in long-haul data transmission over lossy links. A recent study on overlay networks by Liu et al. [1] has shown that breaking a long end-to-end TCP connection into multiple shorter TCP “relay” connections can improve efficiency and fairness. The theoretical results from this study are the basis of our work. The main difference is that we are considering a transparent use of TCP relay nodes inside the network. This means that the end-system does not have to be configured to use overlay networks. From a practical point of view, this is an important aspect as changes in end-system software are difficult to deploy.

Other examples of work show how the use of multiple TCP connections can improve throughput. Split TCP connections are used to cope with differences in the communication media that could cause the congestion control window to close. The idea is to split the connection at the boundary in order to isolate performance issues that are related to a particular medium. This was implemented in I-TCP [2], [3]. Ananth and Duchamp introduce the idea of implementing a single logical end-to-end connection as a series of cascaded TCP connections [4].

B. *TCP Processing on Routers*

In addition to theoretical work on TCP connections, there are numerous examples where some sort of TCP processing has been implemented inside the network (i.e., on a router). A commonly used technique for building application layer firewalls involves inserting a TCP proxy in the communication path of the two communicating end points. Spatscheck et al. show in [5] that TCP connection splicing improves TCP forwarding performance by a factor of two to four as compared to simple IP forwarding on the same hardware.

Layer 4 switches that provide load balancing and transport layer caching functionality often perform TCP traffic redirection. TCP splicing of such redirected connections significantly improves forwarding performance [6], [7] and is an example of simple TCP-level processing of packets on a router.

C. *TCP Processing on Network Processors*

Network processors are embedded system-on-a-chip multiprocessors that are used to implement packet processing functions on the input and output ports of routers. Commercial examples of network processors are the Intel IXP2400 [8], the EZchip NP-1 [9], and the Agere FPP [10]. Network processors have been used in a number of scenarios to provide advanced packet processing functions ranging from simple forwarding [11] to complex functions like network measurement [12].

In the context of TCP processing, network processors have been used to offload TCP processing from high-end server systems [13]. Moving complex TCP processing from end-systems into specialized networking hardware reduces the system load and frees up resources. In our work, the TCP accelerator provides similar functionality as a TCP-offloading system. Instead of providing an interface between the network and a host, the TCP accelerator acts as a connector between two TCP connections inside the network. Packets are buffered and forwarded using a modified TCP state machine.

III. TCP Accelerator Node

A. *System Architecture*

Figure 1 shows the architecture of a TCP acceleration node. The router system implements two forwarding paths for packets. Packets that cannot be accelerated due to resource constraints or non-TCP protocols are forwarded without any modification. In order to identify such packets, it is necessary

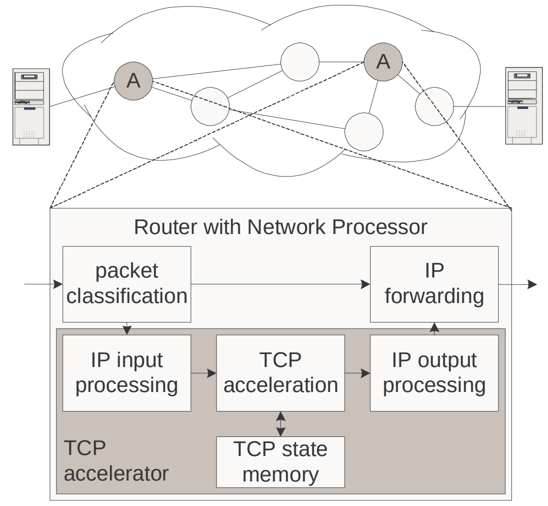

Fig. 1. System Architecture of TCP Acceleration Node.

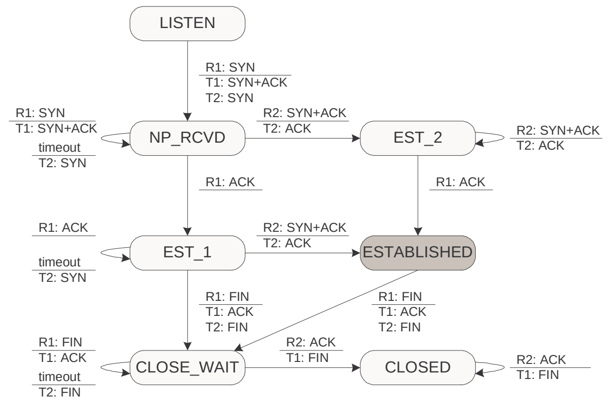

Fig. 2. Connection Establishment and Teardown States of TCP Accelerator for Unidirectional Transfer. R1 indicates reception of data from incoming connection, T2 indicates transmission of data on outgoing connection (T1 and R2 respectively).

to have a packet classification mechanism (e.g., simple 5-tuple hash function). Packets that are accelerated require layer 3 and layer 4 processing, which involves IP input processing, TCP acceleration, and IP output processing. The TCP accelerator has access to a large memory to store TCP state (connection state as well as data buffers).

It is important to note that packets which are processed in the TCP accelerator are not addressed to the router system that performs the acceleration. Instead, the router transparently intercepts these packets and performs the acceleration. The end systems are also unaware of this processing that is performed by the router.

B. *TCP Connection Handling*

The TCP connection management is illustrated by the TCP state machine in Figure 2. It is simpler than a conventional state machine because the possibilities for connection establishment and teardown are limited by the pipeline nature of a TCP accelerator.

The connection setup operates as follows. When the TCP accelerator receives a SYN segment from the requesting end (sender), it responds with SYN/ACK containing its initial sequence number. The accelerator then sends a SYN request segment to the receiving end with another initial sequence number. The connection is in the NP_RCVD state indicating that a connection establishment is in progress. If the TCP accelerator receives the ACK from the sender the connection has been established with the sender and the state is EST_1. Similarly, if the receiver responds first, then the connection establishment is completed by sending an ACK and moving to state EST_2. Eventually, both connections to sender and receiver are established and thus the TCP accelerator moves into the established state.

To handle data packets the TCP accelerator receives a packet, buffers it and acknowledges it to the sender. Then it transmits the data to the receiver (changing the sequence number and possibly even the packet size). It is important to note that by acknowledging the packet to the sender, the TCP accelerator has taken "responsibility" for the packet. Due to the acknowledgement the sender will release the buffer where the packet was stored and future retransmissions will not be possible. The TCP accelerator on the other hand must maintain the copy of the packet until it receives an ACK from the receiver. If no buffer space is available (e.g., due to backlog), the TCP accelerator must not ACK packets from the receiver, thus causing packet loss and a reduction in the congestion window size. Alternatively, it can also proactively advertise a smaller window size (i.e., employing flow control).

For connection termination, the sender sends a FIN as the result of the application issuing a close. When the TCP accelerator receives a FIN, it sends an ACK back to the requesting end and a FIN to the receiving end. From now on, the connection is in CLOSE_WAIT until the TCP accelerator receives an ACK from the receiving side. Then the TCP accelerator sends a FIN to the sender and the connection goes to CLOSED.

C. Processing and Memory Resources

TCP processing requires additional processing and memory resources as compared to plain IP forwarding. The processing consists of IP input and output processing as well as TCP processing. The total processing requirements in terms of the number of RISC instructions executed can be estimated to be around 200 instructions for IP processing [14], 190 instructions for receiving TCP segments, and 210 for transmitting TCP segments [15].

The memory requirements are determined by the size of the TCP connection state (tens of bytes) and the TCP buffer size (tens of kilobytes). The buffer requirements for a TCP accelerator are determined by the maximum window size that is allowed on a connections. The accelerator needs to reliably buffer all packets that have not been acknowledged by the receiver plus all packets that can possible be sent by the sender. Thus, the ideal buffer size is two times the maximum window size of the connection.

Typical network processor systems are equipped with multiple processor cores (four to sixteen) and large SRAM and DRAM memories (1–8MB SRAM, 64–256MB DRAM). Assuming a maximum window size of 64kB, a buffer of 128kB needs to be allocated for maximum efficiency. With this configuration a total of one thousand connections can be supported by 128MB of memory. This is sufficient to support most TCP connections on low-speed edge routers. The additional processing requirement for these connections is well below 1MIPS and thus easily achievable with any current network processor.

IV. TCP Accelerated Networks

When using TCP accelerators in a network, several issues need to be considered.

A. Accelerator Placement

There is one key constraint on the placement of a TCP accelerator. All TCP packets of an accelerated connection need to pass through the accelerator node for it to work properly. Due to the transparency of the accelerator, the end-system is not aware of the placement of the node and thus cannot "force" traffic on a certain route. It is therefore possible that packets take different routes and thus cause incorrect behavior.

Routing changes do happen in networks, but there are topological aspects that can guarantee proper operation if the TCP accelerator is placed appropriately. Many end-systems and subnetworks have only a single access link to the Internet (e.g., hosts connected through access routers or stub-domains). In such a case, there are no options for routing alternatives and traversal of that router is guaranteed. These routers are particularly suitable for TCP acceleration as they match the performance characteristics that can be achieved by an implementation on network processors.

B. Incremental Deployment

A crucial property of any new networking technology is that it can be deployed easily into the existing infrastructure. TCP acceleration is a feature that can improve performance, but is not required for correct network operation. Therefore it is ideal to be deployed incrementally as new routers with network processing capabilities are deployed. It could be conceivable that Internet Service Providers (ISPs) provide acceleration technology initially to their premium customers (which might not exceed the maximum number of accelerated connections on any given router). Also, it is possible to make acceleration available as a best-effort feature, where routers accelerate a TCP connection if buffer memory is available at connection time.

V. Results

We show simulation results to demonstrate that TCP acceleration can effectively increase the throughput of data transfer. The experiments were carried out on the NS-2 [16] simulator. The *FullTcp* agent (with Reno congestion control) was modified to create a new agent that is aware of the

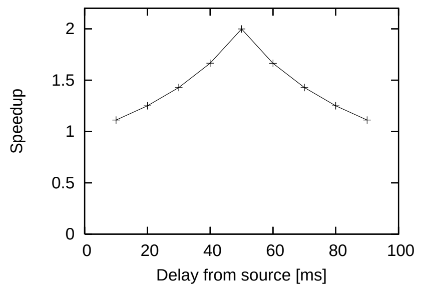

Fig. 3. Speedup of Accelerated TCP over Regular TCP Depending of Location on Node. The x-axis shows the distance of the accelerator from the source with a total end-to-end delay of 100ms.

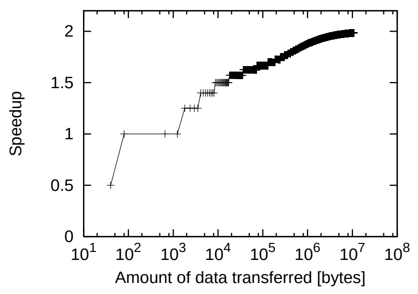

Fig. 4. Speedup of Accelerated TCP over Regular TCP Depending on Connection Duration. The x-axis shows the amount of data that is transferred.

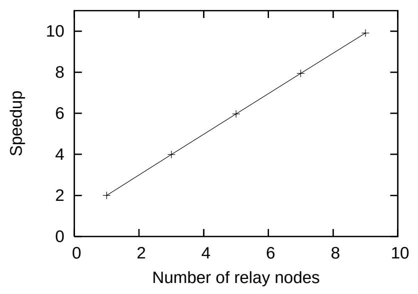

Fig. 5. Speedup of Accelerated TCP over Regular TCP Depending on Number of Accelerators. The x-axis shows the number of evenly spaced acceleration nodes with a total end-to-end delay of 100ms.

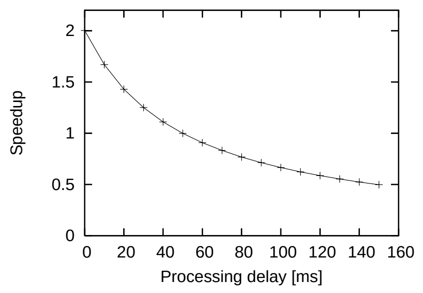

Fig. 6. Speedup of Accelerated TCP over Regular TCP Depending on Processing Delay of Acceleration Node. The x-axis shows the amount of delay introduced due to acceleration processing.

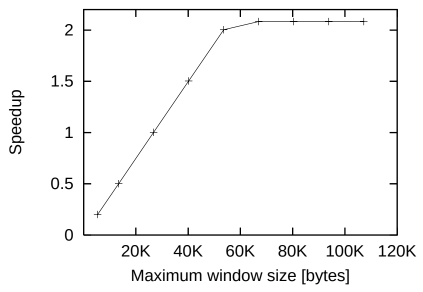

Fig. 7. Speedup of Accelerated TCP over Regular TCP Depending on Maximum Window Size. The x-axis shows the amount of data that can be stored on an acceleration node.

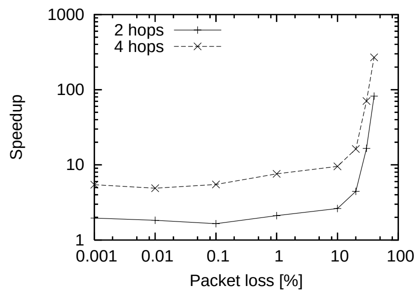

Fig. 8. Speedup of Accelerated TCP over Regular TCP Depending on Packet Loss Rate.

TCP acceleration mechanism. Additionally, a new application module was developed to pull data from the TCP receive buffer and transfer it to the next accelerator node or the end system. This application is used to simulate the functionality of the TCP accelerator node.

In the following subsections, we explore different aspects of TCP acceleration by varying accelerator node placement, introducing packet loss, constraining available processing resources and varying network topology. In each case, we compare the throughput *speedup* offered by TCP acceleration over regular TCP. By "Regular TCP" we mean a standard NS-2 *FullTcp* agent on the same topology. The total end-to-end delay for all experiments is 100ms (unless stated otherwise). Unless explicitly specified, link bandwidths are 100Mb, and the RED queuing discipline is used. The FTP application is used to provide data transfer.

A. Single Relay Node

First, we explore the speedup offered over regular TCP for different placements of a single accelerator node over a link. The placement of the node is varied by changing the distance (delay) of the accelerator node from the source. The results are shown in Figure 3 and show that evenly spaced accelerator nodes are best as they cut the link delay for both sides into half. Unevenly split connections can still benefit from acceleration as the speedup is always greater than 1.

B. Connection Duration

One question that is particularly important in the context of web transfers is the connection duration. Many web documents consist of small objects and TCP connections to download them are often very short. We obtain Figure 4 by exploring the connection duration with a single accelerator node in the middle of the topology. There is some overhead in accelerated TCP only for the initial ACK. Transfers of a few kilobytes already achieve an overall speedup of 1.5. The maximum speedup for for a singe accelerator node is 2 and large file transfers converge to this value.

C. Multiple Acceleration Nodes

The maximum speedup with a single acceleration node is 2. The question arises as to how much speedup multiple acceleration nodes can achieve. The results in Figure 5 assume evenly spaced acceleration nodes totaling an end-to-end delay of 100ms. The maximum speedup increases linearly with the number of nodes. This is due to the reduced round-trip time for each connection, which allows faster acknowledgement of sent data and transmission of the next window. It should be noted that in this case the connection throughput is flow-control-limited, not congestion-control-limited. In the latter case, the speedup would be limited by the available bandwidth.

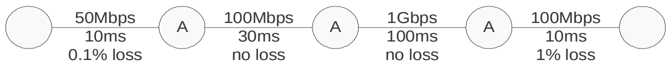

Fig. 9. Topology with Unevenly Spaced Nodes and Different Link Properties.

D. TCP Processing Delay

The previous experiments assume that the processing delay for TCP acceleration is negligible. As we have discussed in Section III-C, TCP acceleration can incur a significant processing cost and it is important to see what impact this has on the achievable speedup. The results for a single accelerator node with varying processing delay are shown in Figure 6. Even with a processing delay of 40ms (which is very large, even for complex packet processing functionality), accelerated TCP still performs better than regular TCP. Processing delays in the lower millisecond range have hardly any impact on the overall performance.

E. Acceleration with Limited Memory

In addition to processing power, TCP acceleration also requires a significant amount of buffer space to be maintained per connection. In Figure 7, we vary the maximum window size on the accelerator node and plot the speedup obtained over regular TCP. The window size is half the amount of data that must be stored on the node in the worst case. The accelerator performs best if the window size is at least as large as the maximum window size used by the sender and receiver. Smaller window sizes linearly decrease the throughput. Larger window sizes doe not improve performance, because the sender cannot utilize the larger window.

F. Acceleration of Lossy Links

One scenario where accelerated TCP significantly outperforms regular TCP is on lossy links (e.g., wireless links). The throughput is higher because local retransmissions from the accelerator node can repair the loss locally and do not require end-to-end retransmissions. Figure 8 shows the speedup of a single accelerator node ("2 hops") and three accelerator nodes ("4 hops") over regular TCP for different packet loss rates. For loss rates around 1–10%, the speedup on a multihop topology is close to tenfold. For higher loss rates (which are unlikely to be encountered in a realistic network) the speedup is even higher.

G. Realistic Topology

Finally, we show the results of one setup that uses a more realistic topology and combines the effects of placement, processing delay, and lossy links. The topology is shown in Figure 9 and a processing delay of 20ms is assumed for all three acceleration nodes. Regular TCP achieves a throughput of 19.1kbps on this topology. Accelerated TCP achieves 72.7kbps, which is equal to a speedup of 3.8. This illustrates that TCP acceleration can provide significant performance improvement in a broad range of network environments, even if acceleration nodes are not placed evenly and even if an additional processing delay is incurred.

VI. Summary

In this work, we have introduced a transparent TCP acceleration technique that can speedup TCP connections without end-system support. We have discussed how such an acceleration system can be implemented on a network processor. The simulation results that we have presented show that a single acceleration node can speedup TCP connection twofold on lossless and more on lossy links. Multiple acceleration nodes can further increase the attainable throughput. As a next step, we plan to implement this system on an Intel IXP2400 system.

Acknowledgements

The authors would like to thank Yong Liu and Yu Gu for providing the NS-2 simulation code used in [1], which has served as the basis for the simulations presented in this paper.

References

[1] Y. Liu, Y. Gu, H. Zhang, W. Gong, and D. Towsley, "Application level relay for high-bandwidth data transport," in *Proc. of First Workshop on Networks for Grid Applications (GridNets)*, San Jose, CA, Oct. 2004.
[2] A. Bakre and B. Badrinath, "I-TCP: Indirect TCP for mobile hosts," in *Proc. of Fifteenth Intl. Conf. on Distributed Computing Systems*, May 1995.
[3] ——, "Implementation and performance evaluation of indirect TCP," *IEEE Transactions on Computers*, pp. 260–278, Mar. 1997.
[4] A. I. Sundararaj and D. Duchamp, "Analytical characterization of the throughput of a split TCP connection," Department of Computer Science, Stevens Institute of Technology," Technical Report, 2003.
[5] O. Spatscheck, J. S. Hansen, J. H. Hartman, and L. L. Peterson, "Optimizing TCP forwarder performance," *IEEE/ACM Transactions on Networking*, pp. 146–157, 2000.
[6] A. Cohen, S. Rangarajan, and H. Slye, "On the performance of TCP splicing for URL-aware redirection," in *USENIX Symposium on Internet Technologies and Systems*, 1999.
[7] G. Apostolopoulos, D. Aubespin, V. Peris, P. Pradhan, and D. Saha, "Design, implementation and performance of a content-based switch," in *Proc. of IEEE INFOCOM 2000*, Tel Aviv, Israel, Mar. 2000, pp. 1117–1126.
[8] *Intel Second Generation Network Processor*, Intel Corporation, 2002, http://www.intel.com/design/network/products/npfamily/ixp2400.htm.
[9] *NP-1 10-Gigabit 7-Layer Network Processor*, EZchip Technologies Ltd., Yokneam, Israel, 2002, http://www.ezchip.com/html/pr_np-1.html.
[10] *PayloadPlus^{TM} Fast Pattern Processor*, Lucent Technologies Inc., Apr. 2000, http://www.agere.com/support/non-nda/docs/FPPProduct-Brief.pdf.
[11] T. Spalink, S. Karlin, L. Peterson, and Y. Gottlieb, "Building a robust software-based router using network processors," in *Proc. of the 18th ACM Symposium on Operating Systems Principles (SOSP)*, Banff, AB, Oct. 2001, pp. 216–229.
[12] R. Ramaswamy, N. Weng, and T. Wolf, "A network processor based passive measurement node," in *Proc. of Passive and Active Measurement Workshop (PAM)*, Boston, MA, Mar. 2005, pp. 337–340, (Extended Abstract).
[13] *Maximizing HP StorageWorks NAS Performance and Efficiency with TCP/IP offload engine (TOE) Accelerated Adapters*, Hewlett-Packard Company, Mar. 2003, http://www.alacritech.com.
[14] R. Ramaswamy, N. Weng, and T. Wolf, "Analysis of network processing workloads," in *Proc. of IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Austin, TX, Mar. 2005, pp. 226–235.
[15] D. D. Clark, V. Jacobson, J. Romkey, and H. Salwen, "An analysis of TCP processing overhead," *IEEE Communications Magazine*, vol. 27, no. 6, pp. 23–29, June 1989.
[16] *The Network Simulator - ns-2*, LBNL, Xerox PARC, UCB, and USC/ISI, http://www.isi.edu/nsnam/ns/.