

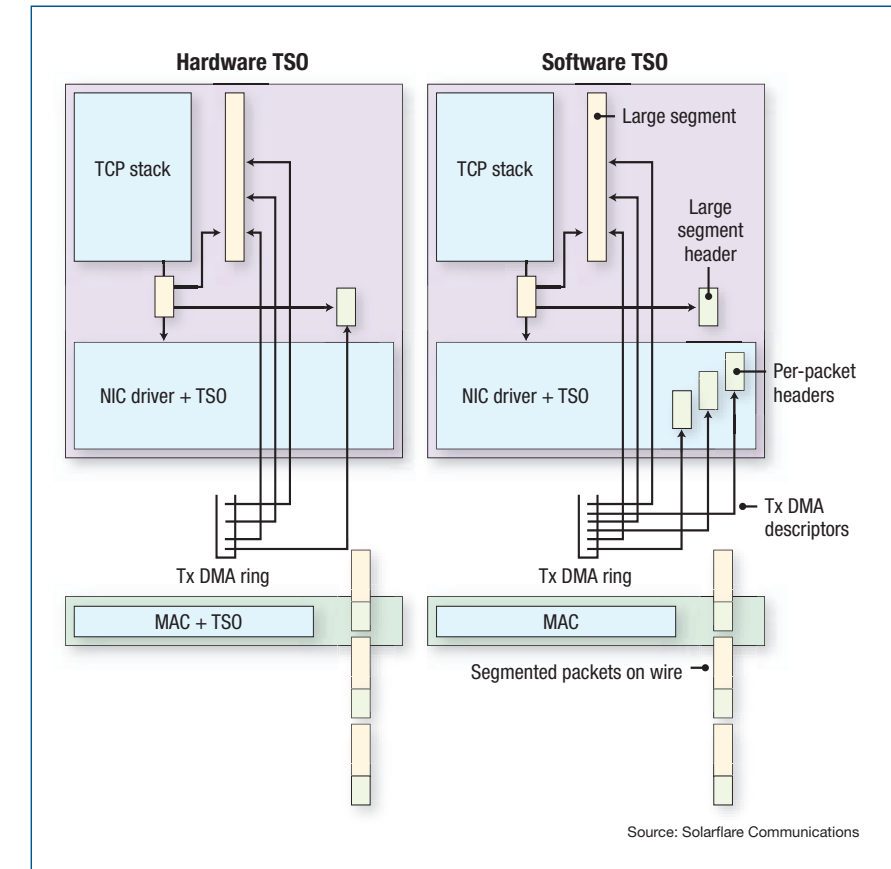
Net-interface accelerators can be a help or hindrance

By Steven Pope
and David Riddoch

Network interface architectures increasingly contain more acceleration features, ranging from checksum validation through to full offload of TCP and higher-layer protocols. Some of these accelerations can be performed efficiently in software. In some circumstances, the use of complex acceleration features can actually harm performance, system integrity and security.

Just as the ubiquitous Ethernet standard has seen many improvements over the past 30 years, the architecture of the host network interface has been subject to great innovation, much of it aimed at enabling operating systems and applications to process data more efficiently at rising link speeds. It is no coincidence that new acceleration features appear when server I/O performance is challenged by an upshift in network link speeds. Some accelerations stand the test of time; others wither against the steady march of CPU and core chip set performance gains. Examining how a given acceleration is used in a system environment gives clues to the feature's long-term utility.

A regular network interface performs no processing of the packets above the link-layer protocol. For example, an Ethernet interface may process the Ethernet frame checksum (FCS) and perform Layer 2 multi-cast filtering. An adapter that can provide optimizations based on the local state contained in the upper-layer protocols of a single frame is defined as a stateless offload adapter. Stateless offloads that can be performed based on higher-level protocols include header splitting as well as TCP/IP



Source: Solarflare Communications

Software-based TCP segmentation offload (TSO) implementations can be almost as efficient as hardware-based implementations.

checksum calculation and verification. Even simple offloads such as checksum can introduce operational conflicts in a system. For example, studies of network error root causes not detected by the Ethernet FCS have found systematic errors in hardware, including in the direct memory access (DMA) controllers within the network interface adapter. Applications are not protected from such errors when checksums are validated in hard-

ware. For that reason, when introducing complex hardware or troubleshooting, it is prudent to disable acceleration features to eliminate them as error sources. As offloads and network adapters become more complex, however, that may cease to be an option. For example, remote DMA protocols enable the network interface to deliver data directly to application buffers. Consequently, extensive changes are required throughout the

software stack. In this model, it is inherent that checksums are generated and validated in the network adapter. Not surprisingly, there are anecdotal reports of (Infiniband) RDMA being disabled throughout a data center in order to solve reliability problems.

Another trade-off to consider with respect to a prospective offload technology is whether the acceleration is best performed by hardware or software. A good example of this is TCP segmentation offload. TSO is a stateless offload in which the TCP layer passes a very large segment (larger than the maximum segment size for the connection) through the stack to the network interface, which is expected to segment it into a number of packets.

Steven Pope is a CTO of Solarflare Communications (Irvine, Calif.).

He holds a PhD in computer science from the University of Cambridge (U.K.).

David Riddoch is chief software architect at Solarflare. He holds a PhD in high-performance networking from Cambridge.



■ about the authors

This method improves performance by reducing per-packet software overheads within the network stack. The segmentation is usually performed by the network adapter and requires complex silicon or an embedded processor to generate headers for each packet. Alternatively, it can be implemented in software at the lowest layer of the stack, as is done in Linux generic segmentation offload (GSO). Contemporary measurements indicate that the performance achieved is almost indistinguishable from that of hardware implementations, but without the silicon cost.

A stateful offload is a network interface acceleration based on the state contained in upper-layer protocols within a sequence of frames. Where TCP is the higher-level protocol, a stateful offload adapter is known as a TCP offload engine (TOE). Because of the inherent requirement to process the higher-layer protocol on the network interface, stateful offloads need a powerful embedded CPU and lots of memory on the adapter. The additional cost and power consumption should be weighed against any benefits of the acceleration.

The TOE also contains a complete implementation of a TCP/IP stack that is distinct from that of the host operating system. That has raised concerns within the OS

community. Our measurements of the TCP conformance of a 10-Gbit/second TOE show it exhibiting lower levels of conformance than the Linux 2.6.9 kernel. In particular, TCP algorithms including selective acknowledgment (SACK) were not implemented by the TOE. That oversight can be expected to degrade performance significantly in production network environments because TCP will respond less efficiently to packet loss.

While each TOE vendor is likely to improve the quality of its TCP stack over time, any integrator that intends to deploy a TOE device should not perform testing based only on a microbenchmark methodology. End users also should expect regular updates to their TOE drivers, firmware and even silicon to fix vulnerabilities and improve conformance.

Hardware-based acceleration should be used wisely. Software-based algorithms usually benefit from Moore's Law, and any acceleration is subject to Amdahl's Law since the scope for application-level acceleration is much less than for microbenchmarks. End users must take care when using stateful offload devices, which may not operate at the same level of conformance or performance as the OS being offloaded. ■



Solarflare Communications, Inc.
9501 Jeronimo Road, Suite 250
Irvine, CA 92618

949.581.6830
949.581.4695 (fax)