Chapter 15

IPsec-Protected Transport of HDTV over IP*

Peter Bellows[1], Jaroslav Flidr[1], Ladan Gharai[1],
Colin Perkins[1], Pawel Chodowiec, and Kris Gaj[2]

[1] *USC Information Sciences Institute, 3811 N. Fairfax Dr. #200, Arlington VA 22203, USA;*
{pbellows|jflidr|ladan|csp}@isi.edu

[2] *Dept. of Electrical and Computer Engineering, George Mason University, 4400 University Drive, Fairfax VA 22030, USA;*
{pchodow1|kgaj}@gmu.edu

Abstract Bandwidth-intensive applications compete directly with the operating system's network stack for CPU cycles. This is particularly true when the stack performs security protocols such as IPsec; the additional load of complex cryptographic transforms overwhelms modern CPUs when data rates exceed 100 Mbps. This paper describes a network-processing accelerator which overcomes these bottlenecks by offloading packet processing and cryptographic transforms to an intelligent interface card. The system achieves sustained 1 Gbps host-to-host bandwidth of encrypted IPsec traffic on commodity CPUs and networks. It appears to the application developer as a normal network interface, because the hardware acceleration is transparent to the user. The system is highly programmable and can support a variety of offload functions. A sample application is described, wherein production-quality HDTV is transported over IP at nearly 900 Mbps, fully secured using IPsec with AES encryption.

15.1 Introduction

As available network bandwidth scales faster than CPU power [Calvin, 2001], the overhead of network protocol processing is becoming increasingly dominant. This means that high-bandwidth applications receive diminishing

*This work is supported by the DARPA Information Technology Office (ITO) as part of the Next Generation Internet program under Grants F30602-00-1-0541 and MDA972-99-C-0022, and by the National Science Foundation under grant 0230738.

P. Lysaght and W. Rosenstiel (eds.),
New Algorithms, Architectures and Applications for Reconfigurable Computing, 183–194.

marginal returns from increases in network performance. The problem is greatly compounded when security is added to the protocol stack. For example, the IP security protocol (IPsec) [ips, 2003] requires complex cryptographic transforms which overwhelm modern CPUs. IPsec benchmarks on current CPUs show maximum throughput of 40–90 Mbps, depending on the encryption used [Fre, 2002]. With 1 Gbps networks now standard and 10 Gbps networks well on their way, the sequential CPU clearly cannot keep up with the load of protocol and security processing. By contrast, application-specific parallel computers such as FPGAs are much better suited to cryptography and other streaming operations. This naturally leads us to consider using dedicated hardware to offload network processing (especially cryptography), so more CPU cycles can be dedicated to the applications which use the data.

This paper describes a prototype of such an offload system, known as "GRIP" (Gigabit-Rate IPsec). The system is a network-processing accelerator card based on Xilinx Virtex FPGAs. GRIP integrates seamlessly into a standard Linux implementation of the TCP/IP/IPsec protocols. It provides full-duplex gigabit-rate acceleration of a variety of operations such as AES, 3DES, SHA-1, SHA-512, and application-specific kernels. To the application developer, all acceleration is completely transparent, and GRIP appears as just another network interface. The hardware is very open and programmable, and can offload processing from various levels of the network stack, while still requiring only a single transfer across the PCI bus. This paper focuses primarily on our efforts to offload the complex cryptographic transforms of IPsec, which, when utilized, are the dominant performance bottleneck of the stack.

As a demonstration of the power of hardware offloading, we have successfully transmitted an encrypted stream of live, production-quality HDTV across a commodity IP network. Video is captured in an HDTV frame-grabber at 890 Mbps, packetized and sent AES-encrypted across the network via a GRIP card. A GRIP card on a receiving machine decrypts the incoming stream, and the video frames are displayed on an HDTV monitor. All video processing is done on the GRIP-enabled machines. In other words, the offloading of the cryptographic transforms frees enough CPU time for substantial video processing with no packet loss on ordinary CPUs (1.3 GHz Pentium III).

This paper describes the hardware, device driver and operating system issues for building the GRIP system and HDTV testbed. We analyze the processing bottlenecks in the accelerated system, and propose enhancements to both the hardware and protocol layers to take the system to the next levels of performance (10 Gbps and beyond).

15.2 GRIP System Architecture

The overall GRIP system is shown in Figure 15.1. It is a combination of an accelerated network interface card, a high-performance device driver, and

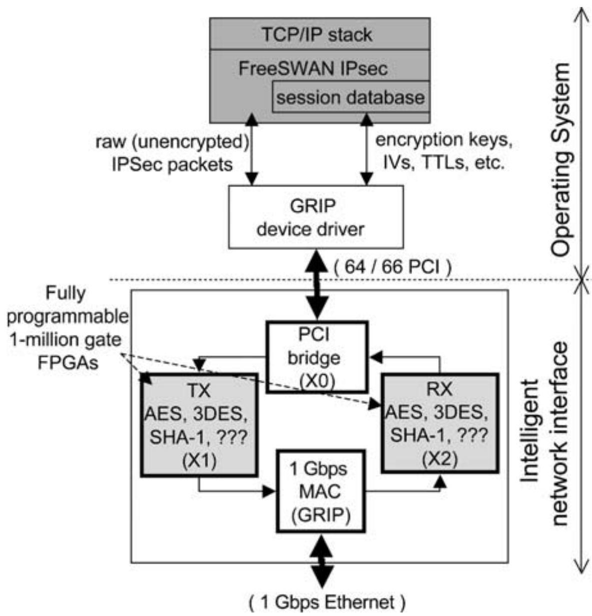

Figure 15.1. GRIP system architecture.

special interactions with the operating system. The interface card is the SLAAC-1V FPGA coprocessor board [Schott et al., 2003] combined with a custom Gigabit Ethernet mezzanine card. The card has a total of four FPGAs which are programmed with network processing functions as follows. One device (X0) acts as a dedicated packet mover/PCI interface, while another (GRIP) provides the interface to the Gigabit Ethernet chipset and common offload functions such as IP checksumming. The remaining two devices (X1 and X2) act as independent transmit and receive processing pipelines, and are fully programmable with any acceleration function. For the HDTV demonstration, X1 and X2 are programmed with AES-128 encryption cores.

The GRIP card interfaces with a normal network stack. The device driver indicates its offload capabilities to the stack, based on the modules that are loaded into X1 and X2. For example in the HDTV application, the driver tells the IPsec layer that accelerated AES encryption is available. This causes IPsec to defer the complex cryptographic transforms to the hardware, passing raw IP/IPsec packets down to the driver with all the appropriate header information but no encryption. The GRIP driver looks up security parameters (key, IV, algorithm, etc.) for the corresponding IPsec session, and prefixes these parameters to each packet before handing it off to the hardware. The X0 device fetches the packet across the PCI bus and passes it to the transmit pipeline (X1). X1 analyzes the packet headers and security prefix, encrypting or providing other security services as specified by the driver. The packet, now completed, is sent to the Ethernet interface on the daughter card. The receive pipeline is just the inverse, passing through the X2 FPGA for decryption. Bottlenecks in other layers of the stack can also be offloaded with this “deferred processing” approach.

15.3 GRIP Hardware

15.3.1 Basic platform

The GRIP hardware platform provides an open, extensible development environment for experimenting with 1 Gbps hardware offload functions. It is based on the SLAAC-1V FPGA board, which was designed for use in a variety of military signal processing applications. SLAAC-1V has three user-programmable Xilinx Virtex 1000 FPGAs (named X0, X1 and X2) connected by separate 72-bit systolic and shared busses. Each FPGA has an estimated 1 million equivalent programmable gates with 32 embedded SRAM banks, and is capable of clock speeds of up to 150 MHz. The FPGAs are connected to 10 independent banks of 1 MB ZBT SRAM, which are independently accessible by the host through passive bus switches. SLAAC-1V also has an on-board flash/SRAM cache for storing FPGA bitstreams, allowing for rapid run-time reconfiguration of the devices. For the GRIP project, we have added a custom Gigabit Ethernet mezzanine card to SLAAC-1V. It has a Vitesse 8840 Media Access Controller (MAC), and a Xilinx Virtex 300 FPGA which interfaces to the X0 chip through a 72-bit connector. The Virtex 300 uses 1 MB of external ZBT SRAM for packet buffering, and performs common offload functions such as filtering and checksumming.

The GRIP platform defines a standard partitioning for packet processing, as described in section 15.2. As described, the X0 and GRIP FPGAs provide a static framework that manages basic packet movement, including the MAC and PCI interfaces. The X0 FPGA contains a packet switch for shuttling packets back and forth between the other FPGAs on the card, and uses a 2-bit framing protocol (“start-of-frame”/“end-of-frame”) to ensure robust synchronization of the data streams. By default, SLAAC-1V has a high-performance DMA engine for mastering the PCI bus. However, PCI transfers for a network interface are small compared to those required for the signal processing applications targeted by SLAAC-1V. Therefore the DMA engine was specially tuned with key features needed for high-rate network-oriented traffic, such as dynamic load balancing, 255-deep scatter-gather tables, programmable interrupt mitigation, and support for misaligned transfers.

With this static framework in place, the X1 and X2 FPGAs are free to be programmed with any packet-processing function desired. To interoperate with the static framework, a packet-processing function simply needs to incorporate a common I/O module and adhere to the 2-bit framing protocol. SLAAC-1V’s ZBT SRAMs are not required by the GRIP infrastructure, leaving them free to be used by packet-processing modules. Note that this partitioning scheme is not ideal in terms of resource utilization—less than half of the circuit resources in X0 and GRIP are currently used. This scheme was chosen because it provides a clean and easily programmable platform for network research. The GRIP hardware platform is further documented in [Bellows et al., 2002].

15.3.2 X1/X2 IPsec Accelerator Cores

A number of packet-processing cores have been developed on the SLAAC-1V/GRIP platform, including AES (Rijndael), 3DES, SHA-1, SHA-512, SNORT-based traffic analysis, rules-based packet filtering (firewall), and intrusion detection [Chodowiec et al., 2001, Grembowski et al., 2002, Hutchings et al., 2002]. For the secure HDTV application, X1 and X2 were loaded with 1 Gb/s AES encryption cores. We chose a space-efficient AES design, which uses a single-stage iterative datapath with inner-round pipelining. The cores support all defined key sizes (128, 192 and 256-bit) and operate in either CBC or counter mode. Because of the non-cyclic nature of counter mode, the counter-mode circuit can maintain maximum throughput for a single stream of data, whereas the CBC-mode circuit requires two interleaved streams for full throughput. For this reason, counter mode was used in the demonstration system.

The AES cores are encapsulated by state machines that read each packet header and any tags prefixed by the device driver, and separate the headers from the payload to be encrypted/decrypted. The details of our AES designs are given in [Chodowiec et al., 2001]. We present FPGA implementation results for the GRIP system in section 15.7.

15.4 Integrating GRIP with the Operating System

The integration of the hardware presented in the section 15.3 is a fairly complex task because, unlike ordinary network cards or crypto-accelerators, GRIP offers potential services to all layers of the OSI architecture. Offloading IPsec—the main focus of current GRIP research—is particularly problematic, as the IPsec stack does not properly reside in any one layer; rather, it could be described as a link-layer component "wrapped" in the IP stack (Figure 15.2). Care must be taken to provide a continuation of services even though parts of higher layers have been offloaded.

For this study we used FreeSWAN, a standard implementation of IPsec for Linux [fre, 2003]. FreeSWAN consists of two main parts: KLIPS and Pluto. KLIPS (KerneL IP Security) contains the Linux kernel patches that implement the IPsec protocols for encryption and authentication. Pluto negotiates the Security Association (SA) parameters for IPsec-protected sockets using the ISAKMP protocol. Figure 15.2 illustrates the integration of FreeSWAN into

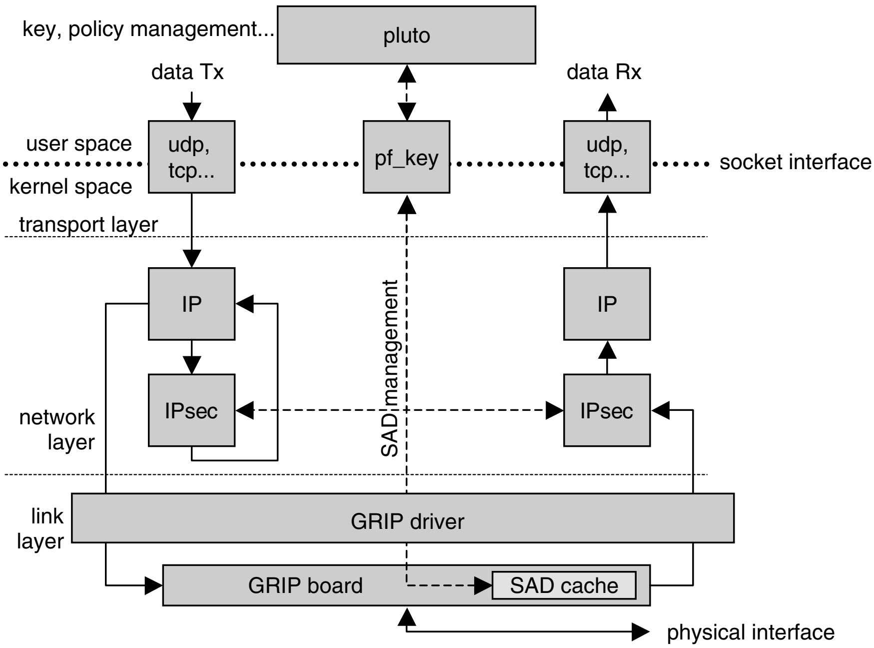

Figure 15.2. The IPsec (FreeSWAN) stack in the kernel architecture.

the system architecture. When Pluto establishes a new SA, it is sent to the IPsec stack via the pf_key socket, where it is stored in the Security Association Database (SAD). At this point, the secure channel is open and ready to go. Any time a packet is sent to an IPsec-protected socket, the IPsec transmit function finds the appropriate SA in the database based on the target IP address, and performs the required cryptographic transforms. After this processing, the packet is returned to the IP layer, which passes it on to the physical interface. The receive mode is the inverse but somewhat less complex. When there are recursive IPsec tunnels, the above process can repeat many times.

In order to accommodate GRIP acceleration we made three modifications to the FreeSWAN IPsec implementation. First, we modified Pluto so that AES Counter mode is the preferred encryption algorithm for negotiating new SA's. Second, we added the ability for the Security Association Database to echo new SA parameters to the appropriate physical interface, using the private space of the corresponding driver. When the GRIP driver receives new SA parameters, it caches encryption keys and other information on the GRIP card for use by the accelerator circuits. Finally, the IPsec transmit and receive functions were slightly modified to support AES counter mode. Any packet associated with an AES SA gets processed as usual—IPsec headers inserted, initialization vectors generated, etc. The only difference is that the packet is passed back to the stack without encrypting the payload. The GRIP driver recognizes these partially-processed packets and tags them with a special prefix that instructs the card to perform the encryption.

15.5 Example Application: Encrypted Transport of HDTV over IP

15.5.1 Background

To demonstrate the performance of the GRIP system, we chose a demanding real-time multimedia application: transport of High Definition Television (HDTV) over IP. Studios and production houses need to transport uncompressed video through various cycles of production, avoiding the artifacts that are an inevitable result of multiple compression cycles. Local transport of uncompressed HDTV between equipment is typically done with the SMPTE-292M standard format for universal exchange [Society of Motion Picture and Television Engineers, 1998]. When production facilities are distributed, the SMPTE-292M signal is typically transported across dedicated fiber connections between sites, but a more economical alternative is desirable. We consider the use of IP networks for this purpose.

15.5.2 Design and Implementation

In previous work [Perkins et al., 2002] we have implemented a system that delivers HDTV over IP networks. The Real-time Transport Protocol (RTP) [Schulzrinne et al., 1996] was chosen as the delivery service. RTP provides media framing, timing recovery and loss detection, to compensate for the inherent unreliability of UDP transport. HDTV capture and playout was via DVS HDstation cards [HDs, 2003], which are connected via SMPTE-292M links to an HDTV camera on the transmitter and an HDTV plasma display on the receiver. These cards were inserted into workstations with standard Gigabit Ethernet cards and dual PCI busses (to reduce contention with the capture/display cards).

The transmitter captures the video data, fragments it to match the network MTU, and adds RTP protocol headers. The native data rate of the video capture is slightly above that of Gigabit Ethernet, so the video capture hardware is programmed to perform color sub-sampling from 10 to 8 bits per component, for a video rate of 890 Mbps. The receiver code takes packets from the network, reassembles video frames, corrects for the effects of network timing jitter, conceals lost packets, and renders the video.

Each video frame is 1.8 million octets in size. To fit within the 9000 octet Gigabit Ethernet MTU, frames are fragmented into approximately 200 RTP packets for transmission. The high packet rates are such that a naive implementation can saturate the memory bandwidth; accordingly, a key design goal is to avoid data copies. We implement scatter send and receive (implemented using the recvfrom() system call with `MSG_PEEK` to read the RTP header, followed by a second call to recvfrom() to read the data) to eliminate data marshalling overheads. Throughput of the system is limited by the interrupt processing and DMA overheads. We observe a linear increase in throughput as the MTU is increased, and require larger than normal MTU to successfully support the full data rate. It is clear that the system operates close to the limit, and that adding IPsec encryption is not feasible without hardware offload.

This HDTV-over-IP testbed served as the benchmark for GRIP performance. The goal was to add line-rate AES cryptography to the system, without any degradation in video throughput and with no special optimizations to the application. Since the GRIP card appears to the system as a standard Ethernet card, it was possible to transparently substitute a GRIP card in place of the normal Ethernet, and run the HDTV application unmodified. The performance results of this test are given in section 15.7.

15.6 Related Work

Two common commercial implementations of cryptographic acceleration are *VPN gateways* and *crypto-accelerators*. The former approach is limited in that it only provides security between LANs with matching hardware (datalink layer security), not end-to-end (network layer) security. The host-based crypto-accelerator reduces the CPU overhead by offloading cryptography, but overwhelms the PCI bus at high data rates. GRIP differs from these approaches in that it is a reprogrammable, full system solution, integrating accelerator hardware into the core operation of the TCP/IP network stack.

A number of other efforts have demonstrated the usefulness of dedicated network processing for accelerating protocol processing or distributed algorithms. Examples of these efforts include HARP [Mummert et al., 1996], Typhoon [Reinhardt et al., 1994], RWCP's GigaE PM project [Sumimoto et al., 1999], and EMP [Shivam et al., 2001]. These efforts rely on embedded processor(s) which do not have sufficient processing power for full-rate offload of complex operations such as AES, and are primarily primarily focused on unidirectional traffic. Other research efforts have integrated FPGAs onto NICs for specific applications such as routing [Lockwood et al., 1997], ATM firewall [McHenry et al., 1997], and distributed FFT [Underwood et al., 2002]. These systems accelerate end applications instead of the network stack, and often lacked the processing power of the GRIP card.

15.7 Results

15.7.1 System Performance

The HDTV demonstration system was built as described in section 15.5.2, with symmetric multiprocessor (SMP) Dell PowerEdge 2500 servers (2x1.3 GHz) running Linux 2.4.18, substituting a GRIP card in place of the standard Ethernet. The full, 890 Mbps HDTV stream was sent with GRIP-accelerated AES encryption and no compression. In addition, we tested for maximum encrypted bandwidth using iperf [ipe, 2003]. Application and operating system bottlenecks were analyzed by running precision profiling tools for 120 second intervals on both the transmitter and receiver. Transmitter and receiver profiling results are comparable, therefore only the transmitter results are presented for brevity. The profiling results are given in Figure 15.3.

The HDTV application achieved full-rate transmission with no packets dropped. Even though the CPU was clearly not overloaded (idle time $> 60\%$!), stress tests such as running other applications simultaneously showed that the system was at the limits of its capabilities. Comparing the SMP and UP cases under iperf, we can see that the only change (after taking into account the 2X factor of available CPU time under SMP) is the amount of idle time. Yet in essence, the performance of the system was unchanged.

To explain these observations, we consider system memory bandwidth. We measured the peak main memory bandwidth of the test system to be 8 Gbps with standard benchmarking tools. This means that in order to sustain gigabit network traffic, each packet can be transferred at most 8 times to/from main memory. We estimate that standard packet-processing will require three memory copies per packet: from the video driver's buffer to the hdtv application buffer, from the application buffer to the network stack, and a copy within the stack to allow IPsec headers to be inserted. The large size of the video buffer inhibits effective caching of the first copy and the read-access of the second copy. This means these copies consume 3 Gbps of main memory bandwidth for 1 Gbps network streams. Three more main memory transfers occur in writing the video frame from the capture card to the system buffer, flushing ready-to-transmit packets from the cache, and reading packets from memory to the GRIP card. In all, we estimate that a 1 Gbps network stream consumes 6 Gbps of main memory bandwidth on this system. Considering that other

Library/Function	bandwidth	idle	kernel	IPsec	grip driver	appplication	libc
HDTV-SMP	893 Mbps	62%	28%	4%	3%	<1%	< 1%
iperf-SMP	989 Mbps	47%	35%	4%	4%	2%	8%
iperf-UP	989 Mbps	0%	70%	9%	4%	3%	12%

Figure 15.3. Transmitter profiling results running the HTDV and iperf applications, showing percentage of CPU time spent in various functions.

Design	*CLB Util.*	*BRAM Util*	*Pred. Perf. (MHz / Gbps)*	*Measured Perf. (MHz / Gbps)*
X0	47%	30%	PCI: 35 / 2.24 I/O: 54 / 1.73	33 / 2.11 33 / 1.06
X1 / X2 (AES)	17%	65%	CORE: 90 / 1.06 I/O: 47 / 1.50	90 / 1.06 33 / 1.06
GRIP	35%	43%	41 / 1.33	33 / 1.06
Other modules:				
3DES	31%	0%	77 / 1.57	83 / 1.69
SHA-1	16%	0%	64 / 1.00	75 / 1.14
SHA-512	23%	6%	50 / 0.62	56 / 0.67

Figure 15.4. Summary of FPGA performance and utilization on Virtex 1000 FPGAs.

system processes are also executing and consuming bandwidth, and that the random nature of network streams likely reduces memory efficiency from the ideal peak performance, we conclude that main memory is indeed the system bottleneck.

15.7.2 Evaluating Hardware Implementations

Results from FPGA circuit implementations are shown in Figure 15.4. As shown in the figure, the static packet-processing infrastructure easily achieves 1 Gbps throughput. Only the AES and SHA cores have low timing margins. Note that there are more than enough resources on SLAAC-1V to combine both AES encryption and a secure hash function at gigabit speeds. Also note that the target technology, the Virtex FPGA family, is five years old; much higher performance could be realized with today's technology.

15.8 Conclusions and Future Work

Network performance is currently doubling every eight months [Calvin, 2001]. Modern CPUs, advancing at the relatively sluggish pace of Moore's Law, are fully consumed by full-rate data at modern line speeds, and completely overwhelmed by full-rate cryptography. This disparity between network bandwidth and CPU power will only worsen as these trends continue. In this paper we have proposed an accelerator architecture that attempts to resolve these bottlenecks now and can scale to higher performance in the future. The unique contributions of this work are not the individual processing modules themselves; for example, 1 Gbps AES encryption has been demonstrated by many others. Rather, we believe the key result is the full system approach to integrating accelerator hardware directly to the network stack itself. The GRIP card is capable of completing packet processing for multiple layers of the stack. This gives a highly efficient coupling to the operating system, with only one pass across the system bus per packet. We have demonstrated this system running at full 1 Gbps line speed with end-to-end encryption on commodity PCs. This provides significant performance improvements over existing implementations of end-to-end IPsec security.

As demonstrated by the HDTV system, this technology is very applicable to signal processing and rich multimedia applications. It could be applied to several new domains of secure applications, such as immersive media (e.g. the collaborative virtual operating room), commercial media distribution, distributed military signal processing, or basic VPNs for high-bandwidth networks.

We would like to investigate other general-purpose offload capabilities on the current platform. A 1 Gbps secure hash core could easily be added to the processing pipelines to give accelerated encryption and authentication simultaneously. More functions could be combined by using the rapid reconfiguration capabilities of SLAAC-1V to switch between a large number of accelerator functions on-demand. Packet sizes obviously make a big difference—larger packets mean less-frequent interrupts. The GRIP system could leverage this by incorporating TCP/IP fragmentation and reassembly, such that PCI bus transfers are larger than what is supported by the physical medium. Finally, several application-specific kernels could be made specifically for accelerating the HDTV system, such as RTP processing and video codecs.

Our results suggest that as we look towards the future and consider ways to scale this technology to multi-gigabit speeds, we must address the limitations of system memory bandwidth. At these speeds, CPU-level caches are of limited use because of the large and random nature of the data streams. While chipset technology improvements help by increasing available bandwidth, performance can also greatly improve by reducing the number of memory copies in the network stack. For a system such as GRIP, three significant improvements are readily available. The first and most beneficial is a direct DMA transfer between the grabber/display card and the GRIP board. The second is the elimination of the extra copy induced by IPsec, by modifying the kernel's network buffer allocation function so that the IPsec headers are accommodated. The third approach is to implement the zero-copy socket interface.

FPGA technology is already capable of multi-gigabit network acceleration. 10-Gbps AES counter mode implementations are straightforward using loop-unrolling [Jarvinen et al., 2003]. Cyclic transforms such as AES CBC mode and SHA will require more aggressive techniques such as more inner-round pipelining, interleaving of data streams, or even multiple units in parallel. We believe that 10 Gbps end-to-end security is possible with emerging commodity system bus (e.g. PCI Express), CPU, and network technologies, using the offload techniques discussed.

References

[Fre, 2002] (2002). *IPsec Performance Benchmarking, http://www.freeswan.org/freeswan_trees/freeswan-1.99/doc/performance.html*. FreeS/WAN.

[HDs, 2003] (2003). *http://www.dvs.de/*. DVS Digital Video Systems.

[fre, 2003] (2003). *http://www.freeswan.org/*. FreeS/Wan.

[ips, 2003] (2003). *Latest RFCs and Internet Drafts for IPsec, http://ietf.org/html.charters/ipsec-charter.html*. IP Security Protocol (IPsec) Charter.

[ipe, 2003] (2003). *Network performance measuring tool, http://dast.nlanr.net/Projects/Iperf/*. National Laboratory for Applied Network Research.

Bellows, P., Flidr, J., Lehman, T., Schott, B., and Underwood, K. D. (2002). GRIP: A reconfigurable architecture for host-based gigabit-rate packet processing. In *Proc. of the IEEE Symposium on Field-Programmable Custom Computing Machines*, Napa Valley, CA.

Calvin, J. (2001). Digital convergence. In *Proceedings of the Workshop on New Visions ofr Large-Scale Networks: Research and Applications*, Vienna, Virginia.

Chodowiec, P., Gaj, K., Bellows, P., and Schott, B. (2001). Experimental testing of the gigabit IPsec-compliant implementations of Rijndael and Triple-DES using SLAAC-1V FPGA accelerator board. In *Proc. of the 4th Int'l Information Security Conf.*, Malaga, Spain.

Grembowski, T., Lien, R., Gaj, K., Nguyen, N., Bellows, P., Flidr, J., Lehman, T., and Schott, B. (2002). Comparative analysis of the hardware implementations of hash functions SHA-1 and SHA-512. In *Proc. of the 5th Int'l Information Security Conf.*, Sao Paulo, Brazil.

Hutchings, B. L., Franklin, R., and Carver, D. (2002). Assisting network intrusion detection with reconfigurable hardware. In *Proc. of the IEEE Symposium on Field-Programmable Custom Computing Machines*, Napa Valley, CA.

Jarvinen, K., Tommiska, M., and Skytta, J. (2003). Fully pipelined memoryless 17.8 Gbps AES-128 encryptor. In *Eleventh ACM International Symposium on Field- Programmable Gate Arrays (FPGA 2003)*, Monterey, California.

Lockwood, J. W., Turner, J. S., and Taylor, D. E. (1997). Field programmable port extender (FPX) for distributed routing and queueing. In *Proc. of the ACM International Symposium on Field Programmable Gate Arrays*, pages 30–39, Napa Valley, CA.

McHenry, J. T., Dowd, P. W., Pellegrino, F. A., Carrozzi, T. M., and Cocks, W. B. (1997). An FPGA-based coprocessor for ATM firewalls. In *Proc. of the IEEE Symposium on FPGAs for Custom Computing Machines*, pages 30–39, Napa Valley, CA.

Mummert, T., Kosak, C., Steenkiste, P., and Fisher, A. (1996). Fine grain parallel communication on general purpose LANs. In *In Proceedings of 1996 International Conference on Supercomputing (ICS96)*, pages 341–349, Philadelphia, PA, USA.

Perkins, C. S., Gharai, L., Lehman, T., and Mankin, A. (2002). Experiments with delivery of HDTV over IP networks. Proc. of the 12th International Packet Video Workshop.

Reinhardt, S. K., Larus, J. R., and Wood, D. A. (1994). Tempest and typhoon: User-level shared memory. In *International Conference on Computer Architecture*, Chicago, Illinois, USA.

Schott, B., Bellows, P., French, M., and Parker, R. (2003). Applications of adaptive computing systems for signal processing challenges. In *Proceedings of the Asia South Pacific Design Automation Conference*, Kitakyushu, Japan.

Schulzrinne, H., Casner, S., Frederick, R., and Jacobson, V. (1996). RTP: A transport protocol for real-time applications. RFC 1889.

Shivam, P., Wyckoff, P., and Panda, D. (2001). EMP: Zero-copy OS-bypass NIC-driven Gigabit Ethernet message passing. In *Proc. of the 2001 Conference on Supercomputing*.

Society of Motion Picture and Television Engineers (1998). Bit-serial digital interface for high-definition television systems. SMPTE-292M.

Sumimoto, S., Tezuka, H., Hori, A., Harada, H., Takahashi, T., and Ishikawa, Y. (1999). The design and evaluation of high performance communication using a Gigabit Ethernet. In *International Conference on Supercomputing*, Rhodes, Greece.

Underwood, K. D., Sass, R. R., and Ligon, W. B. (2002). Analysis of a prototype intelligent network interface. *Concurrency and Computing: Practice and Experience*.