Secure HDTV Transport Over IP Networks

Peter Bellows Jaroslav Flidr Ladan Gharai Colin Perkins
USC Information Sciences Institute
3811 N. Fairfax Dr. #200
Arlington VA 22203, USA
{pbellows|jflidr|ladan|csp}@isi.edu

Abstract

This paper describes an FPGA-based system for IPsec security of high-speed data across commodity IP networks. To demonstrate the system, we have transmitted 890 Mbps raw HDTV video across a commodity network, secured on the fly with the IPsec protocol and AES encryption. Such performance is impossible with software-only implementations, for full line-rate data overwhelms typical CPUs. This is particularly true when cryptographic transforms are required, such as those required by the IP Security (IPsec) protocol. This protocol processing overhead competes directly for CPU cycles against the applications trying to process the high-speed data. We have developed an "intelligent network interface" card based on Xilinx Virtex FPGAs for the purpose of offloading arbitrary protocol processing bottlenecks from the network stack. The network accelerator, named "GRIP" (Gigabit-Rate IPsec), integrates seamlessly into a standard Linux network stack to provide gigabit-rate acceleration of network processing from any of the layers in the stack.

1. Introduction

Bandwidth-intensive applications compete directly with the operating system's network stack for CPU cycles. One such application that we have studied in previous research is the distributed processing of uncompressed, high-resolution video content (HDTV) over commodity IP networks [1]. Increasing the bandwidth available in the network infrastructure theoretically permits the application to process richer, higher-resolution content. However in reality, the application receives diminishing marginal returns from the increased bandwidth, because the overhead of network protocol processing increases proportionally. This means that fewer CPU cycles are available to process even more data. Overall end-to-end performance depends on careful balancing of the throughput capacity for the application, operating system, and network hardware.

This paper describes our efforts to add security features to the HDTV application, using the high-grade encryption of the IP security standard (IPsec) [2]. IPsec adds significant additional overhead to the protocol processing, because of the complexity of the cryptographic transforms required. These transforms overwhelm modern CPUs at high line rates; IPsec benchmarks on current CPUs show maximum throughput of 40-90 Mbps, depending on the encryption algorithm used [3]. 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 constrast, application-specific parallel computers such as field-programmable gate arrays (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), to balance the throughput capacity of the system so that more CPU cycles can be dedicated to the applications which use the data.

We have created a prototype of such a hardware-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.

The GRIP hardware was used to accelerate the IPsec security for the HDTV application. 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 rest of this paper describes the how the components of the system - application, operating system, and hardware - were combined and tuned to provide the high bandwidth. 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).

2. HDTV Transport on Commodity Networks

One example of bandwidth-limited processing is distributed processing of high-resolution video data such as HDTV. We studied how this application could be performed on commodity networks, using hardware offload techniques to maintain a high-level of application throughput.

2.1. Background

High Definition Television (HDTV) is the next generation digital TV standard. It provides several high resolution formats with greater colour depth than standard television signals, using a widescreen 16:9 aspect ratio. The SMPTE-292M standard defines a format for universal exchange of uncompressed HDTV between various types of HDTV equipment (e.g. cameras, encoders, VTRs, editing systems, etc.) [4]. It is a 1.485 Gbps digital serial connection. It is widely used in studios and production houses, allowing HDTV content to be delivered uncompressed through various cycles of production, avoiding the artifacts that are an inevitable result of multiple compression cycles. Once production has been completed, the signal is compressed with MPEG-2 [5, 6] and broadcast at a rate of 19.2Mbps.

This process is effective if production takes place within a single facility. Often, however, production facilities are distributed, and hence there is a need to transport uncompressed content between sites. This is typically achieved by running the SMPTE-292M bit-stream over a dedicated fibre connection, but a more economical alternative is desirable. We consider the use of IP networks for this purpose.

Standards for real-time transport of video over IP networks have reached relative maturity, with the dominant protocol being the Real-time Transport Protocol, RTP [7, 8]. RTP provides media framing, timing recovery and loss detection. It typically runs on UDP/IP networks, inheriting their limitations: unreliable, best effort delivery. RTP applications have developed sophisticated strategies for dealing with timing jitter and packet loss [9]. It is expected that a system for delivery of HDTV over IP will use these to provide a robust service. Receivers use information in the RTP headers to correct for packet loss, and to reconstruct media timing.

2.2. Design and Implementation

The transmitter and receiver are dedicated Dell PowerEdge 2500 servers with dual 1.2GHz Pentium III Xeon processors, running Linux 2.4.18. They were chosen because they have dual PCI bus interfaces: one used for video capture or display, and one for network transmission. Performance on systems with a single PCI bus is poor, due to bus contention and bandwidth limitations. HDTV capture and playout was via DVS HDstation cards [10]. These cards capture HDTV into main memory from a SMPTE-292M link, and regenerate the SMPTE-292M output at the receiver.

The transmitter captures the video data, fragments it to match the network MTU, and adds RTP protocol headers. Video capture and packet assembly are performed in separate threads (due to the blocking nature of the capture API). The native data rate of the captured video signal 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. Both the sender and receiver implement the complete RTP protocol stack.

2.3. Performance Requirements

As noted previously, the video data rate (after colour subsampling) is 890 Mbps. 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 data and packet rates are such that a naive implementation can saturate the memory bandwidth. Accordingly, a key design goal is to avoid data copies, so that the system can support the required data rate. 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. We also offload colour conversion to the display hardware, to avoid extra processing.

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. This is shown in figure 1. Other application optimizations were also required to meet the performance requirements, including tuning of the RTP library. It is clear that the sys-

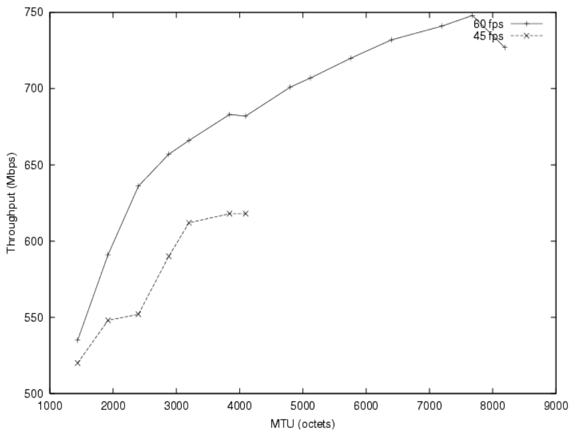


Figure 1. HDTV application throughput for various MTU sizes

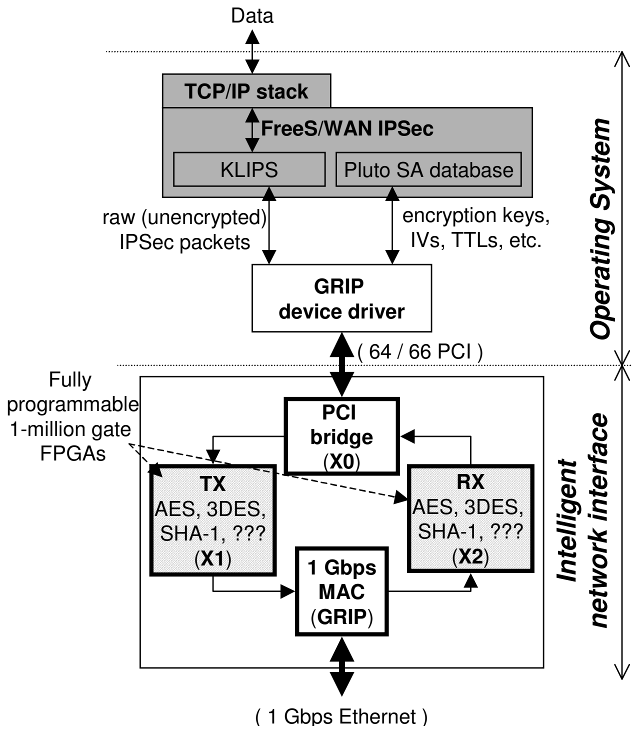
tem is operating close to the limit, and that using IPsec encryption will not be feasible without hardware offload.

3. GRIP System Architecture

In order to address the protocol processing bottleneck, and to make it possible to add IPsec security to the HDTV transport application, we developed the extensible network processing platform known as "GRIP" (Gigabit-Rate IPsec). FPGAs were chosen as the core computational resource, because they are particularly well-suited to highly parallel forms of computation, such as those employed in most cryptographic transforms. This section gives an overview of the GRIP platform, followed by details on the hardware design, firmware modules, and the integration with the operating system.

3.1. GRIP System Overview

The overall GRIP system is diagrammed in figure 2. It is a combination of an accelerated network interface card, a high-performance device driver, and special interactions with the operating system. The interface card is the SLAAC-1V FPGA coprocessor board [11] 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. Because of the programmable nature of the GRIP card, the hardware offload system is very open and extensible, such that various other offload functions can be easily substituted or combined.



Data
TCP/IP stack
FreeS/WAN IPSec
KLIPS
Pluto SA database
raw (unencrypted) IPSec packets
encryption keys, IVs, TTLs, etc.
GRIP device driver
Operating System
(64 / 66 PCI)
Fully programmable 1-million gate FPGAs
PCI bridge (X0)
TX AES, 3DES, SHA-1, ??? (X1)
RX AES, 3DES, SHA-1, ??? (X2)
1 Gbps MAC (GRIP)
Intelligent network interface
(1 Gbps Ethernet)

Figure 2. GRIP system architecture

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 prefixes each packet with a special command header that indicates which offload functions should be performed. 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.

3.2. 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 1 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 3. As described, the X0 and GRIP FPGAs provide a static framework that manages basic packet movement, including the MAC and PCI interfaces, as shown in figure 3. 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 for the GRIP system, the DMA engine was 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,

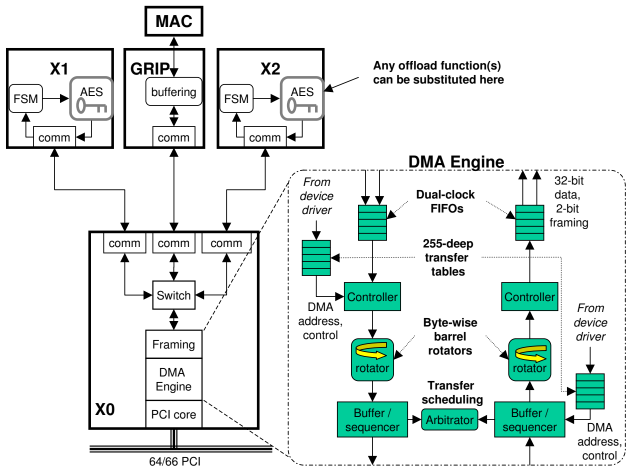


Figure 3. Diagram of GRIP system partitioning.

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 conserving resources - 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 basic GRIP hardware platform is further documented in [12].

3.3. 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 [13, 14, 15]. 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 block diagrams of our AES encryption/decryption units operating in CBC and counter mode are shown in figure 4 a) and b) respectively. The cores support all defined key sizes (128, 192 and 256-bit) and operate in either CBC or counter mode. Both units consist of only two pipeline stages, minimizing the utilization of FPGA resources. Because of the non-cyclic nature of counter mode, the counter-mode circuit can maintain maximum throughput for a single stream

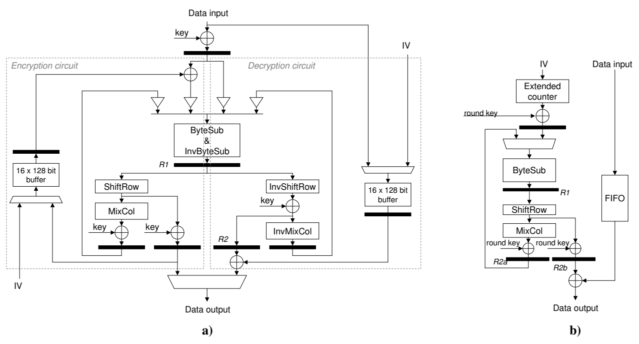


Figure 4. AES encryptor / decryptor in (a) CBC-mode (b) counter-mode.

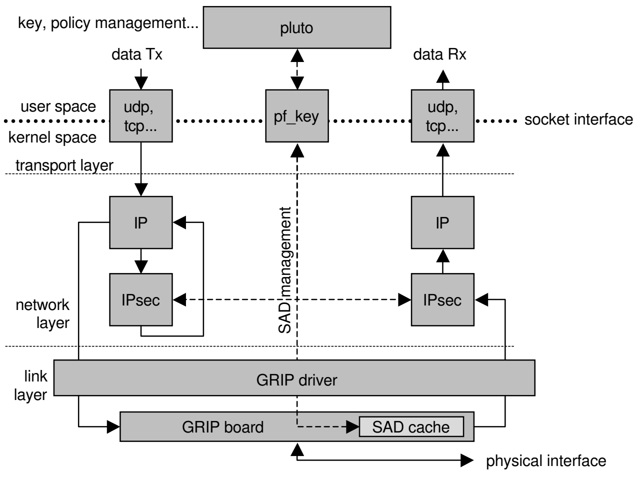


Figure 5. The IPsec (FreeSWAN) stack in the kernel achitecture.

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 [13]. We present FPGA implementation results for the GRIP system in section 4.

3.4. Integrating GRIP with the Operating System

The integration of the hardware presented in the section 3.2 is a fairly complex task because, unlike ordinary network cards or crypto-accelerators, GRIP offers services to three layers of the OSI architecture: the physical, link and network layers. To make a complex matter worse, the IPsec stack - the main focus of current GRIP research - is located in neither the network nor link layers. Rather, it could be described as a link-layer component ”wrapped” in the IP stack (figure 5). Thus 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 [16]. 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. Figure 5 illustrates the integration of FreeSWAN into the system architecture. Pluto negotiates new security associations (SA’s) using the ISAKMP protocol. When a new SA is negotiated, 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. At this point, the secure channel is open and ready to transmit data. The IPsec stack registers itself not only as a network protocol (protocol number 50, [17]) but also as a network device with all its hooks attached to the actual (in our case, GRIP) driver. Thanks to this virtual device, IPsec presents itself as a routable interface. Any time a packet is sent to an IPsec-protected socket, the IPsec transmit function finds the appropriate SA in the database and performs the required cryptographic transforms. After this processing, the packet is handed back to IP which passes it to the physical interface. The receive mode is essentially identical but somewhat less complex. That said, we have to point out that this scenario applies only to the most common cases where a destination address is associated with one SA entry. In more complex cases, such as recursive IPsec tunnels or multiple IPsec interfaces, the above process can repeat many times.

In order to accommodate GRIP acceleration we made three modifications. First, we modified Pluto so that AES Counter mode is the preferred encryption algorithm for negiotiating new SA’s. Second, we altered the actual IPsec stack so that new SA’s are communicated to the GRIP device driver using the driver’s private space. The driver then caches the security parameters (encryption keys, etc.) on the GRIP card for use by the accelerator circuits. Finally, the IPsec transmit and receive functions were slightly modified to produce proper initialization vectors for 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 encryption. The GRIP driver recognizes these partially-processed packets and tags them with a special prefix that instructs the card to perform the encryption, then queues the packets for DMA transfer.

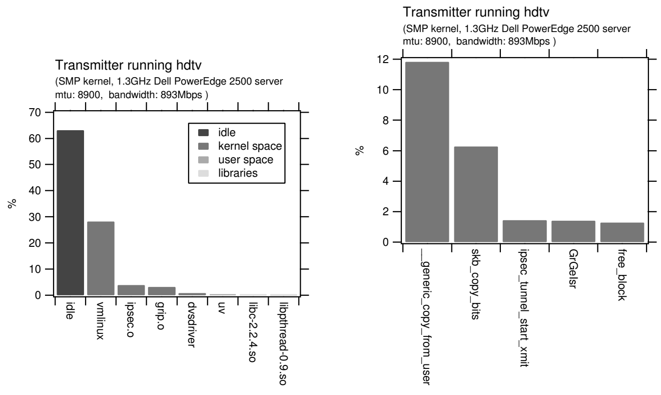
4. Results

4.1. System Performance

In order to characterize the system performance we have set up both symmetric multiprocessor (SMP) and uniprocessor (UP) systems, and measured their performance while running either the iperf application [18] or the HDTV process described in section 2. Each combination was run for 120 seconds, and both the receiver and transmitter were profiled. Transmitter and receiver profiling results are comparable, therefore only the transmitter results are presented for brevity. In both cases the hardware was identical - a Dell PowerEdge 2500 server (2x1.3GHz) with SMP and UP Linux kernels. The profiler used in our experiments was oprofile [19]. Oprofile leverages the hardware performance counters of modern CPUs. Each time the selected counter reaches a preset value, a non-maskable interrupt is issued and a sample taken. The sample indicates which instruction was being executed at the time of the interrupt. The profiler maps these samples ot the symbol table, and constructs a plot of the CPU time spent executing in particular libraries or functions.

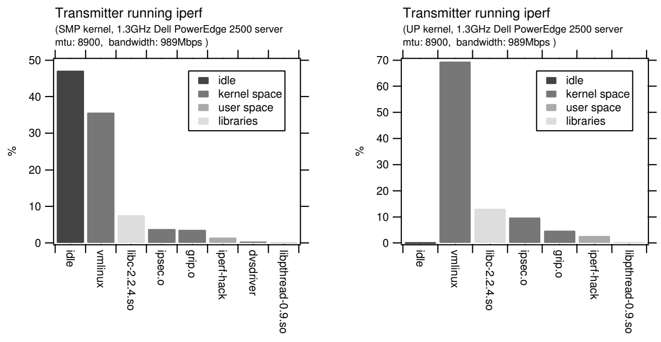
The HDTV application achieved full-rate transmission with no packets dropped. Even though the CPU was clearly not overloaded (idle time $> 60\%$!), stress tests such as running other applications showed that the system was at the limits of its capabilities. Comparing the SMP and UP cases under iperf, 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 transfered 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



(a) **(b)**

Figure 6. Performance of the transmitter running the hdtv application on the SMP kernel. (a) Summary of the CPU time distribution. (b) Detail of the most expensive functions.



(a) **(b)**

Figure 7. Transmitter running iperf on (a) an SMP kernel (b) a UP kernel. In both cases the network throughput is 989 Mbps.

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 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. Also note that memory allocations are two orders of magnitude more expensive in SMP kernels than in UP kernels. All of this explains why the UP system sees only an insignificant decrease in performance even while saturating the CPU.

In summary, these results suggest that while the CPU and system bus resources of current middle and high-end systems are more than sufficient, it is the memory bandwidth which will create a critical bottleneck as demanding applications try to scale to higher network rates. In many of these cases, CPU-level caches are irrelevant, 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 accomodated. The third approach is to implement the zero-copy socket interface. While this is a very efficient modification, it would require significant changes in the kernel code and architecture. Clearly all these improvements are not cumulative - i.e. one cannot reduce the number of required memory copies by three. However any combination of such would reduce it by two and significantly improve the system's performance.

4.2. Evaluating hardware implementations

Results from FPGA circuit implementations are shown in figure 8. As shown in the figure, the static packet-processing infrastructure easily meets the 1 Gbps system bandwidth requirement. These designs should scale with relative ease to multi-gigabit rates by doubling the datapath to 64-bit (currently 32) and circuit optimization. Also note that the Virtex FPGA family is currently five years old, so further speedup could easily be achieved simply by using newer hardware. The most tightly-constrained circuits on the GRIP card are the cryptographic accelerators, as shown. The AES counter mode circuit could easily scale in frequency by partially unrolling the processing loop. Cyclic operations such as AES in CBC mode or SHA would require more aggressive techniques such as more inner-round pipelining and interleaving of data streams. Note that there are more than enough resources on the current system to combine both AES encryption and a secure hash function at gigabit speeds.

5. Related Work

Since IPsec software implementations pose a substantial burden on system resources, hardware accelerators have been in development since the dawn of the IPsec standard. Most approaches could be divided into two categories: *stand-alone, dedicated hardware platforms (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. A few commercial products exist with integrated cryptography on a network interface, such as products from Hifn [20] and Corrent [21]. GRIP differs from these approaches in that it is a reprogrammable, full system solution, integrating accelerator hardware into the core operation of the standard TCP/IP network stack.

A number of other efforts have demonstrated the usefulness of dedicated network processing. These efforts have shown that an embedded processor on the network interface can enhance parallel computing by improving bandwidth or latency or by adding special features. Examples of these efforts include HARP[22], Typhoon[23], RWCP's GigaE PM project[24], and the University of British Columbia's GMS-NP project[25]. Each of these efforts relies on embedded processor(s) and only attempts to achieve peak data rates with unencrypted data in a single direction on the network. The goal of GRIP, by contrast, is to achieve simultaneous bi-directional gigabit throughput with encryption or other complex inline processing. For this, additional processing power is needed on the network interface.

We are aware of two other systems for transport of (mostly) uncompressed HDTV over IP. In conjunction with Tektronix, Inc., we have designed an RTP payload format that allows a SMPTE-292M circuit to be emulated over an IP network [26], using FPGA-based OC48 network hardware. A similar system, using custom network interfaces, has been implemented by NTT labs [27]. Neither system supports IPsec.

Design	*CLB Util.*	*BRAM Util*	*Pred. Perf. (MHz / Gbps)*	*Measured Perf. (MHz / Gbps)*
X0	47%	30%	PCI: 35 / 2.24	33 / 2.11
			I/O: 54 / 1.73	33 / 1.06
X1 / X2 (AES)	17%	65%	CORE: 90 / 1.06	90 / 1.06
			I/O: 47 / 1.50	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%	70 / 0.93	81 / 1.04

Figure 8. Summary of FPGA performance and utilization on Virtex 1000 FPGAs

6. Conclusions and future work

This HDTV broadcast application is representative of a class of bandwidth-limited distributed processing problems, which could theoretically benefit from greater available bandwidth but are hampered by the operating system overhead of marshalling all the data. Network performance is currently doubling every eight months [28]. Modern CPUs, advancing at the relatively sluggish pace of Moore's Law, are falling further and further behind in their ability to supply full line-rate data to such applications, most notably when cryptographic security protocols are used. 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.

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. Future work in implementing the main memory bandwidth optimizations suggested in section 4 could potentially have the greatest impact on end-to-end system bandwidth.

FPGA technology is already capable of multi-gigabit network acceleration. 10-Gbps AES counter mode implementations are straightforward using loop-unrolling [29]. 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.

7. Acknowledgements

This work was 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.

References

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

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

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

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

[5] ISO/IEC: Generic coding of moving pictures and associated audio information: Video (1996) ISO/IEC 13818-2.

[6] ISO/IEC: Generic coding of moving pictures and associated audio information: Systems (1996) ISO/IEC 13818-1.

[7] Schulzrinne, H.: RTP profile for audio and video conferences with minimal control (1996) RFC 1890.

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

[9] Perkins, C.S., Hodson, O., Hardman, V.: A survey of packet loss recovery techniques for streaming media. IEEE Network Magazine (1998)

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

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

[12] Bellows, P., Flidr, J., Lehman, T., Schott, B., Underwood, K.D.: 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 (2002)

[13] Chodowiec, P., Gaj, K., Bellows, P., Schott, B.: 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 (2001)

[14] Grembowski, T., Lien, R., Gaj, K., Nguyen, N., Bellows, P., Flidr, J., Lehman, T., Schott, B.: 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 (2002)

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

[16] FreeS/Wan: http://www.freeswan.org/. (2003)

[17] Kent, S., Atkinson, R.: RFC2401: Security architecture for the internet protocol (1998)

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

[19] John Levon: System-wide profiler for Linux x86 systems (2003) http://oprofile.sourceforge.net/.

[20] Hifn Corporation: Corporate website, http://www.hifn.com. (2003)

[21] Corrent Corporation: Corporate website, http://www.corrent.com. (2003)

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

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

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

[25] Coady, Y., Ong, J.S., Feeley, M.J.: Using embedded network processors to implement global memory management in a workstation cluster. In: Proc. of The Eighth IEEE International Symposium on High Performance Distributed Computing, Redondo Beach, California, USA (1999)

[26] Gharai, L., Perkins, C.S., Goncher, G., Mankin, A.: RTP payload format for society of motion picture and television engineers (SMPTE) 292M video. Internet Engineering Task Force (2003) RFC 3497.

[27] NTT Innovation Laboratories: Uncompressed HDTV transmission system over the internet. NTT Press Release (2001) http://www.ntt.co.jp/news/news01e/-0110/011026.html.

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

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