January 2002

The Influence of ATM on Operating Systems

Jonathan M. Smith
University of Pennsylvania, jms@cis.upenn.edu

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-02-29.

The Influence of ATM on Operating Systems

Abstract

The features of ATM offered many attractions to the application community, such as fine-grained multiplexing and high-throughput links. These created considerable challenges for the O.S. designer, since a small protocol data unit size (the 48 byte "cell") and link bandwidths within a (binary) order of magnitude of memory bandwidths demanded considerable rethinking of operating system structure.

Using an historical and personal perspective, this paper describes two aspects of that rethinking which I participated in directly, namely, those of new event signaling and memory buffering schemes. Ideas and techniques stemming from ATM network research influenced first research operating systems and then commercial operating systems. The positive results of ATM networking, although indirect, have benefited applications and systems far beyond the original design goals.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-02-29.

The Influence of ATM on Operating Systems

Jonathan M. Smith*
University of Pennsylvania
jms@cis.upenn.edu

Abstract

The features of ATM offered many attractions to the application community, such as fine-grained multiplexing and high-throughput links. These created considerable challenges for the O.S. designer, since a small protocol data unit size (the 48 byte "cell") and link bandwidths within a (binary) order of magnitude of memory bandwidths demanded considerable rethinking of operating system structure.

Using an historical and personal perspective, this paper describes two aspects of that rethinking which I participated in directly, namely, those of new event signalling and memory buffering schemes. Ideas and techniques stemming from ATM network research influenced first research operating systems and then commercial operating systems. The positive results of ATM networking, although indirect, have benefitted applications and systems far beyond the original design goals.

1 Introduction

The various technical contributions of the US "Gigabit Testbed" program[Computer 90] were both manifold and undersold. Amongst the more significant contributions were those made in the AURORA Gigabit Testbed, which linked Penn, Bellcore, IBM Research and MIT in the Northeast corridor of the United States [Clark 93]. While AURORA experimented with two packet formats, "Packet Transfer Mode" (PTM) and "Asynchronous Transfer Mode" (ATM), in retrospect the ATM-centric research had significantly more impact on modern operating system software.

It is is worthwhile to set the stage. While AURORA was initiated in 1990 under the HPCA ("Gore Bill") the research was actually kicked off at Penn and MIT in 1989 under Project DAWN, funded by Bellcore. The goal was to build an integrated LAN/WAN infrastructure using an OC-48 SONET channel provided by Bell Atlantic, MCI and Nynex. Using a clever OC-12 cross-connect topology devised by Dave Sincoskie of Bellcore, we were able to concurrently operate PTM over SONET from Penn to IBM and MIT, and ATM over SONET from Penn to Bellcore to MIT. While supercomputers were used in several other testbeds, AURORA focused on workstation technology, as it was believed (correctly, in retrospect) that delivering high throughputs to this class of machine would have more long-term impact. The available workstation technology was the first generation of RISC computers, which were characterized by an increased number of simpler instructions executed per clock cycle, higher clock rates, and relatively poor DRAM and I/O throughputs given the computational performance. As it is today, multiple levels of cache memory were used to resolve some of the CPU/DRAM performance "gap" [Patterson 02].

One major research direction pursued at Penn was an architectural study of the challenges in delivering the substantial bandwidths (while not so impressive today, the OC-3c bandwidths we targeted to in 1989/1990 were over 13 times as fast as the commonly used LAN technology). This architectural study took the form of a hardware/software codesign, where a hardware subsystem, in the form of an ATM host interface [Traw 91, Traw 93], was designed in concert with new operating system support [Smith 90, Smith 93, Chung 95] designed to expose the strengths of the ATM technology, in particular its substantial bandwidth. While a secondary consideration at the time, another lasting contribution came from exposing the fine-grained control of multiplexing possible with ATM [Nahrstedt 96].

*Work supported by the NSF and ARPA under Cooperative Agreement NCR-8919038 with CNRI, by Bell Communications Research under Project DAWN, by an IBM Faculty Development Award, and by the Hewlett-Packard Corporation.

The remainder of this paper is organized in three sections. The next section, on "Buffer Management", discusses some of the challenges associated with memory architectures in the early 1990s and connects a thread of research results which led to changes in many commercial operating systems. Section 3 covers changes in event-signalling architectures, which have had considerable influence on the research community but perhaps less on the commercial front. Finally, in Section 4 we conclude the paper, observing first that ATM-driven advances in software paved the way for Fast Ethernet and faster LANs, and second that the same fundamental stresses amongst trendlines are still in place as network engineers and workstation designers continue to march to different drummers.

2 Buffer Management

In early 1990, I began thinking hard about the shape of an operating system appropriate for the support of a workstation in an environment with very high performance networks. David Farber, my colleague at the University of Pennsylvania, had worked with Bob Kahn to get the Gigabit Testbed initiative underway. Kahn had set a "Gigabit litmus test", which demanded a gigabit per second throughput, delivered to or from *a single application*.

While it was quite clear even then that a supercomputer [Borman 89] could deliver or accommodate such throughputs, it was also clear that workstations would be extremely challenged if they were called upon to meet the "litmus test". Even if the technical challenges of the host adaptor hardware were met, it was unclear if the workstation could handle the data flow from the adaptor. While the merits of the RISC workstation technology were clear from a computational perspective, it was unclear how they would handle data management, particularly in the performance regimes the ATM technology required.

I laid out principles for a new operating system called "UPWARDS" [Smith 90] (for "U Penn Wide Area Research Distributed System") intended for a network with bandwidth within an order of magnitude of memory bandwidth.

These included (quoting from the paper):

- UPWARDS scheduling is entirely synchronous; the only "interrupt" allowed is that of the system clock driving the scheduler. Interrupts complicate device drivers and are really an artifact of a desire for high levels of multiprocessing, rather than high performance computing with low levels of multiprogramming. In addition, interrupts defeat architectural features necessary for very high performance, since as caches and pipelines.
- The UPWARDS interprocess communication mechanism is shared memory.

While like many operating systems designs, UPWARDS was never fully realized, it provided considerable guidance in what was to follow. Others were of course working in this space, in particular Clark and Tennenhouse [Clark 90] had likewise noted the importance of memory bandwidth, with their focus being on protocol architectures.

2.1 Hardware Context, O.S. Presumptions and Basic Arithmetic

Working with IBM gave AURORA researchers access to a particularly potent workstation technology, the RS/6000, which was just being made available at the time the testbeds were forming. The IBM RS/6000 had been recently introduced [Bakoglu 90] and was IBM's entry in the engineering workstation market. All indications were that it was quite competitive [Simmons 90], being equal to supercomputers one generation old (on computational workloads).

The outstanding feature of the RS/6000 was its relatively large memory bandwidth (using a tool we developed, we measured a Model 320 to have 1 Gbit/s of bandwidth, the Model 530 to have 1.8 Gbit/s of memory bandwidth, and the Model 580 to have 2.5 Gbit/s of memory bandwidth). Others confirmed this later with benchmarks [McVoy 96]. The I/O bus technology, the Microchannel, was an apparent limitation. We chose to work around this by applying a specialized style of direct-memory-access transfer called streaming, which provided twice the throughput of normal DMA, giving us some hope of reaching the application performance targets for AURORA. The Microchannel throughput in this mode was 320 Mbit/s, adequate for the OC-3c data rate of the adaptor.

Looking at some basic performance metrics for this machine, it appeared to be a reasonable platform, but it was clear that some new thinking was required: the off-the-shelf operating system (AIX) would have to be modified, both to reduce copying and to support multimedia. I began to examine how the UPWARDS ideas could be implemented using AIX (IBM's version of UNIX) as a starting point on the RS/6000.

2.2 Reducing Copying

Even with the Gbit/s range memory throughputs on the RS/6000, it was clear [Watson 87] that reducing buffer copying would have considerable impact on performance. We avoided some software-driven copying by use of a hardware feature of the IBM RS/6000, a set of Translation Control Words (TCWs) that provided virtual addressing capability to I/O devices such as our host adaptor. The challenge of structuring memory and choosing where it was addressed for adaptor transfers, as well as scheduling transfers in such as manner as to allow maximum concurrency with other processing, such as that required by the application.

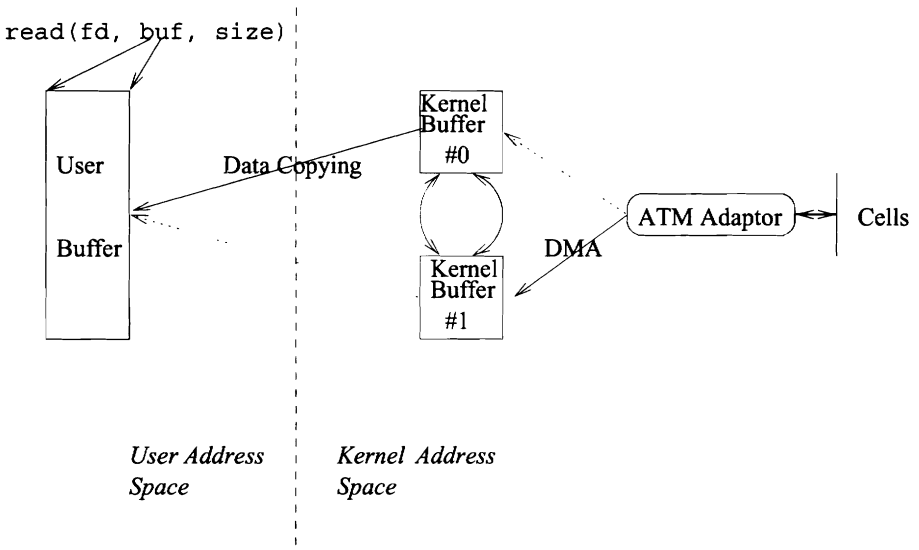

Figure 1: Kernel Buffer Ring - Concurrency with Copying

Careful management of a pair of buffers (this is a fairly standard scheme for network adaptor device drivers, sometimes generalized as a buffer ring), as shown in Figure 1, gave us considerable throughput, and considerable concurrency, but the performance of the system for small buffers (a more common case than the 32 or 64KB "jumbo-grams" we and others used for evaluating performance...) degraded significantly.

Recoding of the driver support for the UNIX `read()/write()` calls allowed transfers directly to or from a user buffer passed to a system call, as shown in Figure 2. This essentially achieved "zero-copy" operation, since only the required DMA from the adaptor was required to get the data into application memory.

Of course, the application was blocked while this occurred, so there was little concurrent activity for the process using the buffer. Thus we sought to remove the system call overhead for small packets by implementing a shared memory region between kernel and user space, with a carefully orchestrated passing of control of the shared regions between user process and kernel to ensure that all available concurrent operation of the hardware could be exploited. The architecture is illustrated in Figure 3.

While all of these configurations were implemented and were able to fully utilize the ATM link at the larger packet sizes, the shared memory interface far outperformed the system-call based scheme on small packets, had the least effect on other applications, and fully exploited the ability of the bus-mastering adaptor to operate concurrently with the processor. It also demonstrated that the basic intuitions of the initial UPWARDS were experimentally sound.

These implementations were used to support applications ranging from a wide-area distributed shared memory to a telerobotics system. The applications used a TCP/IP [Alexander 94] implemented using a virtual device abstraction we used to cope with the fact that the ATM adaptor was implemented as *two* Microchannel cards, one for ATM segmentation, with the other for ATM reassembly.

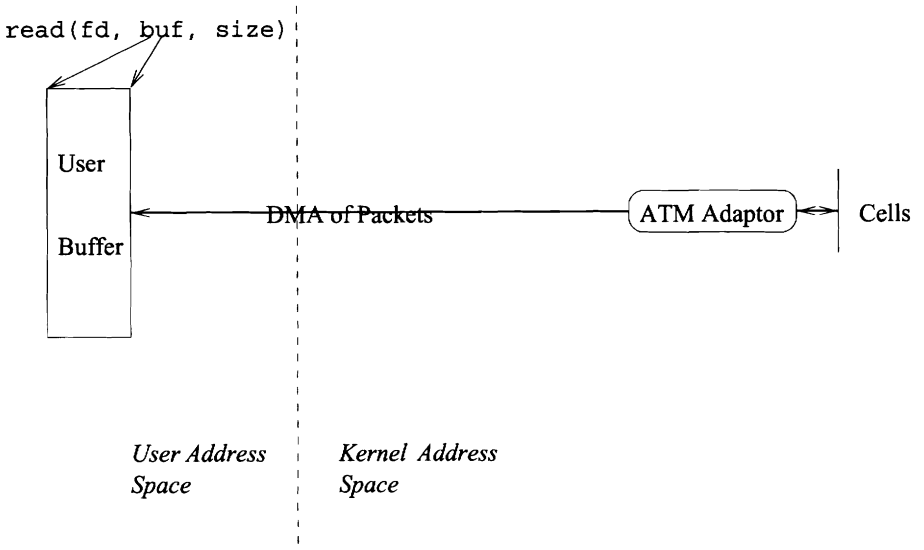

Figure 2: No copy (other than required DMA), *but* no concurrency

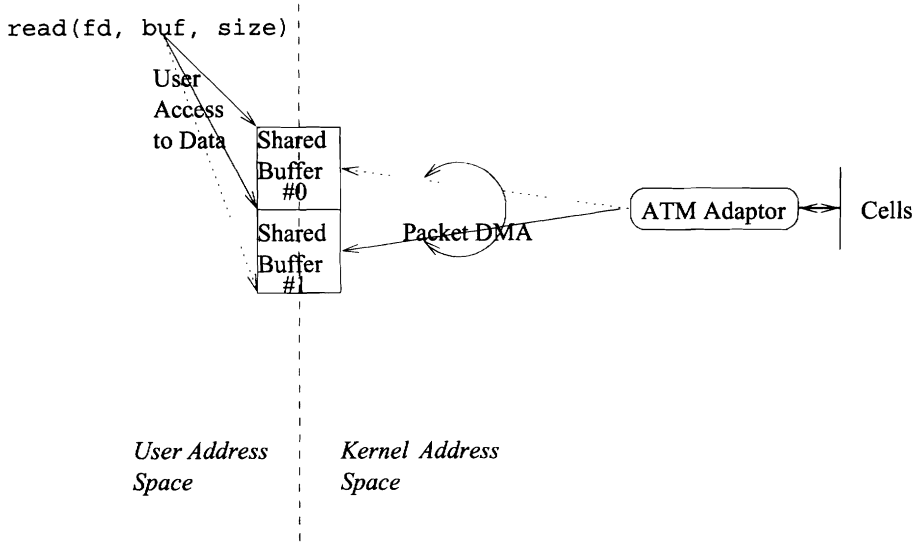

Figure 3: Memory buffers shared between user and kernel address spaces

2.3 Related and Follow-On Work; Commercial Impact

The most important derivative work in the research community was that of Peterson and Druschel. In a paper presented in Tucson in 1992 [Traw 92], we discussed the basic double-buffering strategy which allowed application processing of data in one buffer while allowing a streaming transfer into another buffer. Druschel [Druschel 93] adopted this basic strategy in his "fbuf" buffer management architecture, but rather than using VM protection coupled to double-buffering, Druschel used the type system of his implementation language to control access to the buffers. Absent extra mechanism, the buffering schemes we developed could be made even more efficient, as Druschel demonstrated

in his implementation of fbufs for Bruce Davie's Osiris ATM adaptor. While the end-to-end throughput of Druschel's system was limited by an upper bound on the TurboCHANNEL write throughput to about 300 Mbps, adaptor to host memory transfers were demonstrated at over 500 Mbps with UDP packets [Druschel 94]. As far as I am aware, TCP throughput was never demonstrated with this system; my understanding is that a flawed DMA controller made a repeatable measurement difficult. Nonetheless, fbufs are a nice improvement over the basic shared memory abstraction which started with UPWARDS. A refinement of this architecture was developed in LRP [Druschel 96].

The most important work done concurrently was by the Hewlett-Packard Laboratories [Dalton 93, Banks 93, Edwards 94] in Bristol, UK. There, a high performance host adaptor architecture was developed which allowed a direct mapping of substantial buffer memory on the adaptor directly into application address space. The power of this approach was that it was then reasonably straightforward to write an application-level TCP stack which directly manipulated buffer memory on the adaptor, minimizing copying across the bus to which the adaptor was attached. In many ways, this user-level TCP/IP set directions for the user-implemented protocols in vertically structured operating systems such as the exokernel [Engler 95].

Commercial vendors paid considerable attention to the software architectures we developed. One example is the work of Chu [Chu 96], who adapted the shared memory techniques and fbufs to the commercial Solaris operating system supplied with Sun Microsystems computers. The Chu work supported a 622 Mbit/s ATM adaptor developed by Sun.

3 Event Signalling

Event-signalling within the network subsystem between the hardware network interface device and the software device driver is typically accomplished via polling or device-generated interrupts. In our implementation of an OC-3c ATM host interface for the IBM RS/6000 family of workstations [Traw 93, Smith 93], we replaced the traditional forms of this crucial function with "clocked interrupts." Clocked interrupts, like polling, examine the state of the network interface to observe events which require host operations to be performed. Unlike polling, which requires a thread of execution to continually examine the network interface's state, clocked interrupts perform this examination periodically upon the expiration of a fine-granularity timer. In comparison to interrupts, clock interrupts are generated indirectly by the timer and not directly by the state change event.

3.1 Hardware/Software Context, O.S. Presumptions and Basic Arithmetic

Consider a system with an interrupt service overhead of C seconds per interrupt, and k active channels, each with events arriving at an average rate of λ events per second. Independent of interrupt service, each event costs α seconds to service, e.g., to transfer the data from the device. The offered traffic is $\lambda * k$, and in a system based on an interrupt-per-event, the total overhead will be $\lambda * k * (C + \alpha)$. Since the maximum number of events serviced per second will be $1/C + \alpha$, the relationship between parameters is that $1 > \lambda * k * (C + \alpha)$. Assuming that C and α are for the most part fixed, we can increase the number of active channels and reduce the arrival rate on each, or we can increase the arrival rate and decrease the number of active channels.

However, for clocked interrupts delivered at a rate β per second, the capacity limit is $1 > \beta * C + \lambda * k * \alpha$. Since α is very small for small units such as characters, and C is very large, it makes sense to use clocked interrupts, especially when a reasonable value of β can be employed. In the case of modern workstations, C is about a millisecond. Note that as the traffic level rises, more work is done on each clock "tick," so that the data transfer rate $\lambda * k * \alpha$ asymptotically bounds the system performance, rather than the interrupt service rate. We note that traditional interrupt service schemes can be improved, e.g., by aggregating traffic into larger packets (this reduces λ significantly, while typically causing a slight increase in α), by using an interrupt on one channel to prompt scanning of other channels, or masking interrupts and polling some traffic intensity threshold.

One would therefore expect that for application workloads characterized by high throughput, heavy multiplexing, and/or "real-time" traffic, clocked interrupts should be more effective than either traditional polling or interrupts.

3.2 Clocked Interrupts

The ATM host adaptor we developed for the IBM RS/6000 [Traw 93] was designed with clocked-interrupt based event-signalling in mind. One artifact of this was that there was no support for interrupt generation, an omission that would plague us later, when we were evaluating whether the clocked interrupt scheme performed as we had intended and needed comparison against interrupts to make the case. While a comparative evaluation would have strengthened the technical case for clocked interrupts, we were able to demonstrate two key assertions. First, clocked interrupts could deliver throughputs up to the hardware limits of the RS/6000, in this case slightly over 130 Mbit/s on the Model 580 processor. Second, clocked interrupts provided excellent support for applications with multimedia streams; this was demonstrated by teleoperating [Nahrstedt 96] a robot arm over an ATM network using several RS/6000s.

The question of whether any advantage actually accrued to clocked interrupts intrigued us, so we set out to study it in a second generation of our host interface architecture. The second generation combined segmentation and reassembly in a single board, and absorbed other useful features from the design of the Afterburner [Banks 93, Dalton 93]. It also operated at OC-12c, or four times the link rate of the RS/6000 adaptor which operated at the OC-3c link rate of 155 Mbit/s.

To get tight control of the clock on the PA-Risc we altered the clock support subsystem to perform clock division, and used the basic single-copy TCP/IP stack developed by HP Bristol [Edwards 94, Edwards 95]. We were then able to construct a polling based scheme using a user process, apply traditional interrupts, and operate clocked interrupts at a variety of clock rates. We used a 640+ SONET compliant link to interconnect two HP 9000 Model 755s back-to-back, and then performed experiments using the `netperf` [IND 95] performance analysis tool, with machines both unloaded (Figure 4) and loaded with a computationally intensive workload (Figure 5).

While the experimental setup and results here are covered in [Chung 95] and [Smith 99], we have the data graphically in this paper rather than in the previous tabular format. For these two diagrams, the label "Poll" with a rate indicates the clocked interrupt service rate, that is, how many times per second the device status is checked. 0.4Khz means that the device is checked 400 times per second. There is considerable variability in performance, and this is relatively more important for small buffer sizes, as the performance for large buffers will be dominated by data transfer limits.

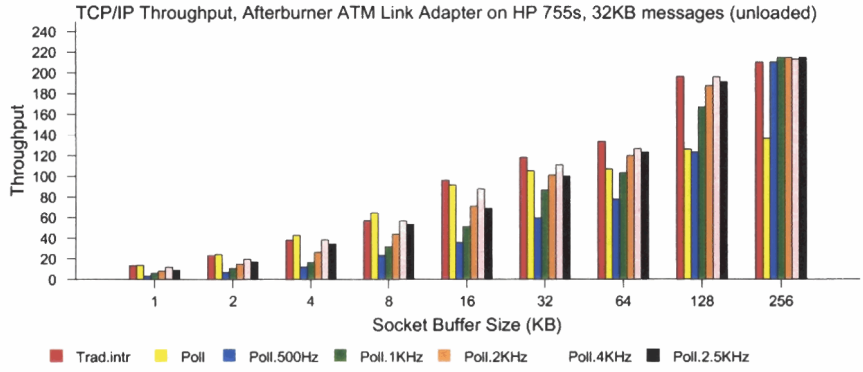

Figure 4: Throughput versus clocking, unloaded machine

It is clear from these results that at high polling rates, the clocked interrupt scheme is able to keep up with the traditional interrupt scheme, which is almost everywhere the best performer, with the exception of polling, which does best for small packet sizes. In a lightly-loaded environment, interrupts would appear to be the best solution, except for some anomalous, but repeatable results which show polling best for small socket buffer sizes.

Another important factor in networking performance, and perhaps a more important parameter for distributed applications is the round-trip latency induced by the software supporting the adaptor. Since the hardware was a constant, we could directly compare the software overheads of the three schemes. This was done with the following

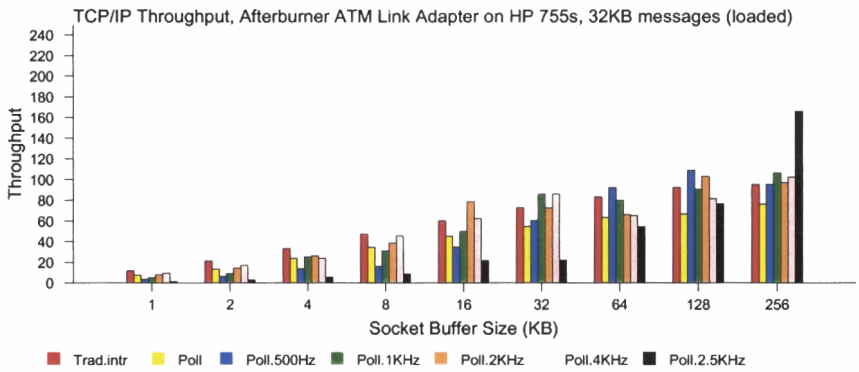

Figure 5: Throughput versus clocking, loaded machine

test. An artificial network load was created using `netperf` with a socket buffer size of 262144 bytes and operating it continuously. Against this background load, ICMP ECHO packets of 4K bytes were sent to the TCP/IP receiver, which was where the event-signaling performance differences would be evident. Sixty tests were done to remove anomalies. Our results showed that traditional interrupts and clocked interrupts at 500 Hz performed similarly, yielding minimum, average and worst case times of 5/12/18 ms, and 4/11/25 ms, respectively. When the systems were not loaded, the performances were 3/3/3 ms and 4/4/6 ms. This suggests that clocked interrupts performed slightly better under heavy load, but slightly worse under unloaded conditions.

We draw three conclusions about this event-signalling architecture. First, clocked interrupts can provide throughput equivalent to the best throughput available from traditional interrupts; both methods provide better performance than polling as implemented here. Second, clocked interrupts provide higher throughput when the processor is loaded by a computationally-intensive process; this suggests that clocked interrupts may be a viable mechanism for heavily loaded systems such as servers, which might also suffer from Ramakrishnan's *receive livelock*. Third, clocked interrupts provide better round-trip delay performance when systems are heavily loaded for large ICMP ECHO packets.

3.3 Related and Follow-On Work; Commercial Impact

The work by Ramakrishnan [Ramakrishnan 93] on "receive livelock" pointed out a real problem with O.S. management of interrupts on fast network adaptors. While Ramakrishnan studied FDDI, the basic problems of an operating system entering a completely interrupt-driven state were applicable to ATM, and where SAR was not done in hardware (as in first generation ATM host interfaces from Fore Systems, which did SAR in software), could be far worse than for FDDI. Mogul and Ramakrishnan [Mogul 97] noted that clocked interrupts were immune from such considerations, but developed a hybrid architecture, which was interrupt-driven under light load, but switched to polling when a certain threshold of load was passed.

A large and influential body of work at the University of Cambridge [Black 95, Roscoe 95, Hyden 94, Leslie 96] was driven by the need to support multimedia. The resulting *Nemesis* operating system had many features, including Black's scheduling scheme [Black 95] and its architecture as a single address space operating system [Roscoe 95] which enabled high-performance. In many ways this operating system was well-suited to the fine-grained resource multiplexing of which ATM was capable. This is not surprising, as a variety of ATM technologies had their origin at Cambridge in, or slightly before, this period.

The most important derivative work was done by Peter Druschel. In his work on Soft Timers [Aron 00], he confirmed that clock-driven activity was a powerful paradigm for networking subsystems. In a heavily multiplexed subsystem, Morris [Morris 99] demonstrated the value of a polling-like scheme, supporting the basic analysis of traditional interrupts versus polling we presented above. The Click router has had direct commercial impact, as it has

been productized by Mazu Networks, Inc.

4 Conclusions

In our discussion of changes in operating system technology, we noted that RISC architectures relied heavily on caches to overcome mismatches in speed between CPUs and DRAMs. Buffer copying tracks DRAM performance rather than base processor performance. In the network attachment arena, the small ATM cell size of *ca.* 50 bytes took about 3 microseconds to arrive at 130 Mbps; 1500 bytes arrives in 12 microseconds at 1 Gbps and 1.2 microseconds at 10 Gbps, or within a rough order of magnitude of the figures with ATM in the early 1990s.

We have shown in this paper how operating systems were revamped to meet the challenges posed by the ATM networks of the early 1990s, and as the paragraph above illustrates, those challenges have not retreated but have in some sense regrouped and reappeared. At the same time, the use of the Internet to bear various forms of multimedia, such as voice and streaming media, has again required operating systems [Muir 01] to examine scheduling and event signalling, both for processing of network traffic and for delivery of that traffic to host applications.

If there is a single lesson to take home from this paper, it is that the ATM technology served as a vanguard to force a rethinking of operating system architecture that might otherwise have been delayed until today, and while operating systems continue to evolve, the effects of this rethinking have already been felt.

5 Acknowledgments

I appreciate the great work of my graduate students Brendan Traw, Jeffrey Chung, Klara Nahrstedt, Michael Massa and Scott Alexander. I learned a good deal about hardware working with Brendan Traw and Bruce Davie; I hope they learned some roughly equivalent amount about system software. I apologize in advance to anyone whose work I left out; the paper was not intended as a comprehensive survey.

References

[Alexander 94] D. Scott Alexander, C. Brendan S. Traw and Jonathan M. Smith, "Embedding High Speed ATM in UNIX IP" *USENIX High-Speed Networking Symposium*, Oakland, CA pp. 119-121 (August 1994)

[Aron 00] M. Aron and P. Druschel, "Soft timers: efficient microsecond software timer support for network processing", *ACM TOCS* 18(3), pp. 197-228 (2000).

[Bakoglu 90] H. B. Bakoglu, G. F. Grohoski and R. K. Montoye, "The IBM RISC System/6000 processor: Hardware overview", *IBM Journal of Research and Development*, 34(1), pp. 12-22 (January, 1990).

[Banks 93] D. Banks and M. Prudence, "A High-Performance Network Architecture for a PA-RISC Workstation," *IEEE JSAC*, 11(2), pp. 191-202 (Feb. 1993).

[Black 95] R. J. Black, "Explicit Network Scheduling", Ph.D. Thesis, University of Cambridge, (April 1995).

[Borman 89] D. A. Borman, "Implementing TCP/IP on a Cray Computer", *ACM Computer Communications Review*, 19(2), pp. 11-15, (April 1989).

[Chu 96] J. Chu, "Zero-Copy TCP in Solaris", *Proceedings, 1996 USENIX*, pp. 253-264 (January 1996).

[Chung 95] J. D. Chung, C. B. S. Traw and J. M. Smith, "Event-Signaling within Higher-Performance Network Subsystems," *Proceedings, HPCS '95*, Mystic, CT, pp. 220-225 (August 1995).

[Clark 90] D. D. Clark and D. L. Tennenhouse, "Architectural Considerations for a New Generation of Protocols", *Proc. SIGCOMM 1990*, Philadelphia, PA (September 1990).

[Clark 93] David D. Clark, Bruce S. Davie, David J. Farber, Inder S. Gopal, Bharath K. Kadaba, W. David Sincoskie, Jonathan M. Smith, and David L. Tennenhouse, "The AURORA Gigabit Testbed," *Computer Networks and ISDN Systems* 25(6), pp. 599-621, North-Holland (January 1993).

[Computer 90] IEEE Computer Staff, "Gigabit Network Testbeds," *IEEE Computer*, 23(9), pp. 77-80 (September 1990).

[Dalton 93] C. Dalton et al., "Afterburner: A network-independent card provides architectural support for high-performance protocols," *IEEE Network*, pp. 36-43 (July 1993).

[Druschel 93] Peter Druschel and Larry Peterson, "Fbufs: A high-bandwidth cross-domain data transfer facility", *Proc. SOSP*, 1993.

[Druschel 94] Peter Druschel, Larry Peterson and Bruce Davie, "Experiences with a High-speed Network Adaptor: A Software Perspective", *Proc. ACM SIGCOMM*, pp. 2-13, (August-September 1994).

[Druschel 96] Peter Druschel and Gaurav Banga, "Lazy Receiver Processing (LRP): A Network Subsystem Architecture for Server Systems", *Proc. OSDI*, pp. 261-275, 1996.

[Edwards 94] A. Edwards, G. Watson, J. Lumley, D. Banks, C. Calamvokis and C. Dalton, "User-space protocols deliver high performance to applications on a low-cost Gb/s LAN," in *Proceedings, 1994 SIGCOMM Conference*, London, UK, 1994.

[Edwards 95] Aled Edwards and Steve Muir, "Experiences implementing a high-performance TCP in user-space", in *Proceedings, ACM SIGCOMM Conference*, Cambridge, MA, pp. 196-205, (August-September 1995),

[Engler 95] D. Engler, M. F. Kaashoek, and J. O'Toole, "Exokernel: An Operating System Architecture for Application-Level Resource Management", *Proc. SOSP*, December 1995.

[Hyden 94] E. Hyden, "Operating System Support for Quality of Service", Ph.D. Thesis, University of Cambridge (February 1994).

[IND 95] Hewlett-Packard Information Networks Division, "Netperf: A Network Performance Benchmark (Revision 2.0)", February 15, 1995.

[Leslie 96] Ian M. Leslie, Derek McAuley, Richard Black, Timothy Roscoe, Paul T. Barham, David Evers, Robin Fairbairns and Eoin Hyden, "The Design and Implementation of an Operating System to Support Distributed Multimedia Applications", *IEEE Journal of Selected Areas in Communications*, 14(7), pp. 1280-1297, (1996).

[McVoy 96] Larry McVoy and Carl Staelin, "*lmbench*: Portable tools for Performance Analysis", *Proceedings, 1996 USENIX Conference*, San Diego, CA, pp. 279-294 (January 1996).

[Mogul 97] J. Mogul and K. K. Ramakrishnan, "Eliminating Receive Livelock in an Interrupt-Driven Kernel", *ACM TOCS*, 15(3), pp. 217-252, 1997.

[Morris 99] R. Morris, E. Kohler, J. Jannotti and M. F. Kaashoek, "The Click Modular Router", *Proc. SOSP*, pp. 217-231 (1999).

[Muir 01] Steve Muir, "Piglet: An Operating System for Network Appliances", Ph.D. Thesis, CIS Department, University of Pennsylvania, 2001.

[Nahrstedt 96] K. Nahrstedt and J. M. Smith, "Design, Implementation and Experiences of the OMEGA End-Point Architecture", *IEEE JSAC* 14(7), pp. 1263-1279 (September 1996).

[Patterson 02] D. A. Patterson and J. L. Hennessy, "Computer Architecture: A Quantitative Approach (3d Ed.)", Morgan Kauffman, 2002.

[Ramakrishnan 93] K. K. Ramakrishnan, "Performance Considerations in Designing Network Interfaces," *IEEE JSAC* 11(2), pp. 203-219 (Feb. 1993).

[Roscoe 95] T. Roscoe, "The Structure of a Multi-Service Operating System", Ph.D. Thesis, University of Cambridge (August 1995).

[Simmons 90] M. L. Simmons and H. J. Wasserman, "Los Alamos Experiences with the IBM RS/6000 Workstation", Technical Report LA-11831-MS (March 1990).

[Smith 90] Jonathan M. Smith, "UPWARDS Operating System: Goals and Relevance to Supercomputing," *Lawrence Livermore Laboratories/IDA Supercomputing Research Center Workshop on Supercomputer Operating Systems*, Livermore, CA (July 10-12 1990).

[Smith 93] Jonathan M. Smith and C. Brendan S. Traw, "Giving Applications Access to Gb/s Networking," *IEEE Network* 7(4), pp. 44-52, (July 1993).

[Smith 99] J. M. Smith, J. D. Chung and C. B. S. Traw, "Interrupts", *Encyclopedia of Electrical and Electronics Engineering*, Volume 10, Wiley(1999), pp. 667-673.

[Traw 91] C. Brendan S. Traw and Jonathan M. Smith, "A High Performance ATM Host Interface for ATM Networks", *Proceedings, ACM SIGCOMM Conference*, Zurich, SWITZERLAND, pp. 317-325 (September 1991).

[Traw 92] C. Brendan S. Traw and Jonathan M. Smith, "Implementation and Performance of an ATM Host Interface for Workstations", *Proceedings, IEEE Workshop on the Architecture and Implementation of High-Performance Communications Subsystems (HPCS '92)*, Tucson, AZ, pp. 101-104 (February 1992).

[Traw 93] C. Brendan S. Traw and Jonathan M. Smith, "Hardware/Software Organization of a High-Performance ATM Host Interface," *IEEE JSAC* 11(2), pp. 240-253 (Feb. 1993).

[Watson 87] Richard W. Watson and Sandy A. Mamrak, "Gaining Efficiency in Transport Services by Appropriate Design and Implementation Choices" *ACM Transactions on Computer Systems*, 5(2), pp. 97-120 (May 1987).