ACTIVE Interconnects: Let's have some guts!

Jonathan M. Smith, Ilija Hadžić and William S. Marcus*
University of Pennsylvania and Bellcore

Abstract

Active networks provide on the fly user-configurability of network infrastructures. One consequence is that more of the computation/communications continuum is available to the systems designer. The design space of active networking can be dimensioned by (1) usability, (2) flexibility, (3) security, and (4) performance. Given the voracious appetite for network bandwidth now apparent, performance is a major concern in realizing the active networks concept. These concerns may be ill-founded, as they are focused on a narrow region of the design space for active network infrastructures. This (admittedly wildly speculative!) paper explores a wider set of possibilities.

1 Introduction

"Active Networks" [SFG+96, TSS+97] is a term used to describe post-Internet programmable network infrastructures which allow users to customize the network from the edges. The most aggressive proposals (e.g., Tennenhouse's *capsules* [TSS+97]) promote per-packet programmability of the infrastructure, in effect treating packets as programs. Capsules are one instance of what can be done with the "Store-Translate-and-Forward" (STF) model developed by Smith in 1993, and first applied with David Feldmeier in the design of "Protocol Boosters." [FMS+98]. In the STF model, the translation point is where programming takes place (given the equivalence between language recognition and computing) and distinguishes STF networks from Store-and-Forward packet switching.

Operational infrastructures have been produced by several initial efforts such as Active Bridging [ASNS97], ANTS [WGT98] and PLAN [HKM+]. These efforts are points in a design space which has many dimensions, the most important of which are flexibility, security, usability and performance.

*This work was supported by DARPA under Contracts #DABT63-95-C-0073 and #N66001-96-C-852, with additional support from the Intel Corporation.

1.1 The Active Networks Design Space

The four key factors in an Active Networking system are *usability*, *flexibility*, *security*, and *performance*. In the SwitchWare[1] project at Penn and Bellcore, we use these four factors to define a design space. The majority of initial points in this design space have been software systems, *e.g.*, the specialized Programming Language for Active Networks (PLAN) [PLA], the restrictions of a general purpose programming language in the ALIEN active loader [ASNS97], and the secure bootstrap and trust extensions of the Secure Active Network Environment (SANE) [AAKS98]. Other efforts (e.g., MIT's ANTS [WGT98]) occupy additional areas in the space.

Usability tends to be an aesthetic issue, and has to be evaluated with psychological means. Flexibility has some of the same aspects, but can at least grossly be evaluated by the capability or capacity of the language recognizer. Thus, we would expect a regular-expression pattern matcher to be less flexible than a Turing machine. Security is a tricky business [AAKS98], if for no other reason than it is ultimately rather application-specific. Performance is easily quantified (*e.g.*, as bits per second of throughput or packets per second of forwarding capacity), and the additional complexity required for making a network configurable is likely to reduce performance under these measures.

1.2 The easy way out: Avoiding the Fast Path

One way to add flexibility without reducing forwarding performance is to remove the flexibility from the *transport plane* and restrict it to the *control plane*. Figure 1 illustrates a continuum defined by computing a ratio of computation / bandwidth, or *COB*.

A representative home computer, *e.g.*, a 266 Mhz Intel Pentium, might execute about 500 peak MIPS. Measuring throughput in Mb/s, for the home network, COB will be $(5 * 10^8)/(5 * 10^4)$, or about 10^4. This suggests that there are about 10,000 instructions available for each bit of data passing through the system.

"Active" functions performed on networks with smaller COB values must be very simple, or accelerated in some way for per-packet processing (for example by examining only headers, the locus of processing complexity, rather than entire packets [CJRS89]). Today, with fast forwarding technologies operating at multi-Gb/s line rates, the arguments for operating in the control plane are clear. This has led to architectures for separating control and transport planes in ATM networks [HR98, vdML98] and in the Internet [SAM$^+$98].

This control/transport separation is already performed in an implicit way in an IP Switch [NML98]; once a flow has been detected, the switch controller operates independently of the switched flows. The separation is made explicit in the Multigigabit router [PCB$^+$98] developed by BBN, which uses a custom switch fabric internally. The BBN router has separate cards for forwarding and routing, and migrates many address lookups out into line cards rather than centralizing them. The fast-path assumption is that the lookups will take place out of Alpha 21164 L1 or L2 caches; when they don't (*e.g.*, ARP required) the router's behavior deviates slightly from traditional routers.

[1] http://www.cis.upenn.edu/~switchware

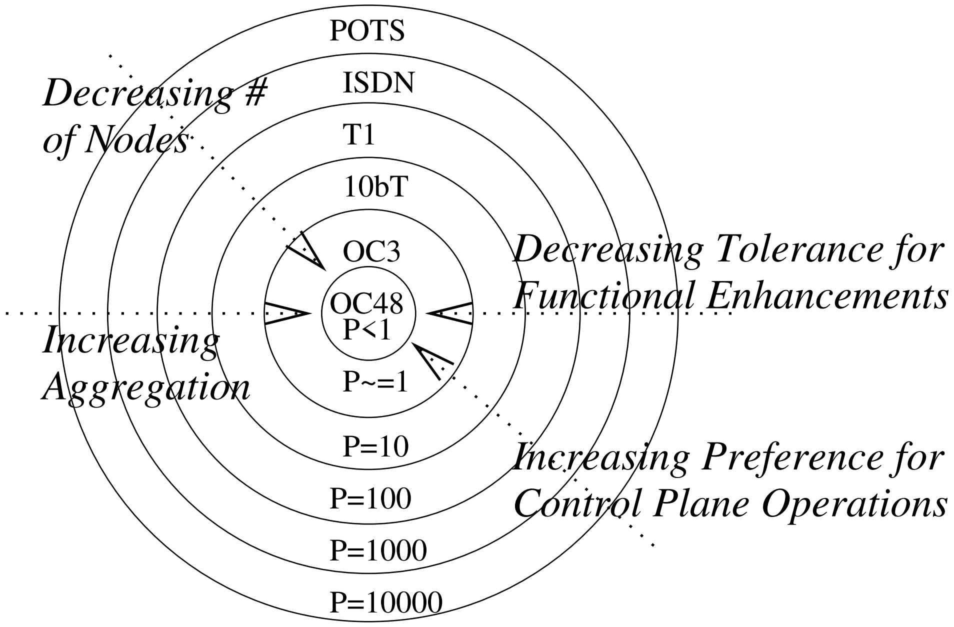

Figure 1: The COB ring model

2 Getting on the Fast Path: The P4

We can operate in an active fashion at any performance point, by an appropriate reduction along one or more of the other axes in the design space. This is clearly true, as a *reductio ad absurdum* is that to do nothing costs nothing in security, performance, usability and flexibility. The intellectually interesting question is whether we can gain some increment in flexibility and still operate in the highest performance regimes.

One exploratory effort in this direction is Hadzic's Programmable Protocol Processing Pipeline [HS97], (or P4). The P4 consists of four FIFO-connected Altera 8000 PLDs with some auxiliary logic. A small crossbar is used to virtualize ordering of operations on the input stream. An input block is used to split ATM cell data from headers, and an output block used to recombine them. The P4 sacrifices some flexibility and usability as a consequence of limited gates in the PLDs, but is able to operate on ATM cells at OC3 line rates without particularly aggressive clock speeds. A high-level block diagram of the system is shown in Figure 2.

At present, FEC and decoders have been implemented, and the P4 has demonstrated that it can adaptively mask errors on a link, and that its presence can greatly enhance TCP/IP performance in the face of errors.

Of particular note is the P4's ability to be reconfigured on-the-fly, via commands and loads issued by an out-of-path controller. The multiplicity of FPGAs and the crossbar allow preloading of functions, which can be "switched-in" to the transport path extremely rapidly (orders of magnitude faster than the time to load an FPGA). The P4 thus sheds some light into a heretofore unexplored region of the Active Nets design space.

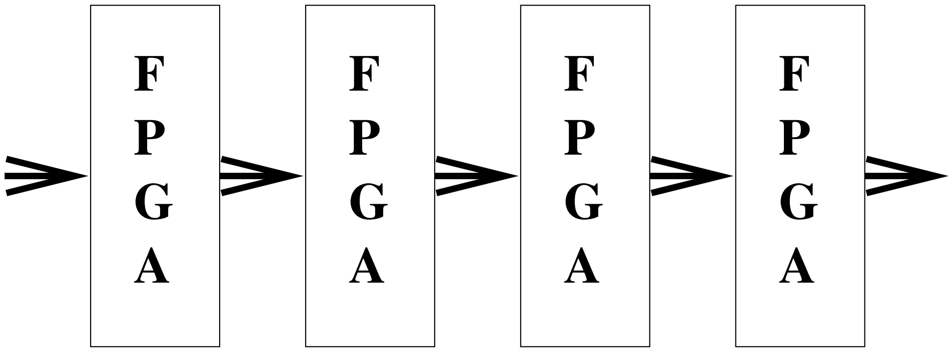

Figure 2: High-level view of Hadzic's Programmable Protocol Processing Pipeline (P4)

3 Let's Have some Guts – and some Fun!

This section is entirely "blue-sky", essentially focusing on a few issues which are Active Networks desiderata, and hence warrant further investigation, to ensure having hardware ready for accelerating network evolution. This hardware will form the "guts" of many future active networking infrastructures. Lessons learned in its construction will be useful at a number of architectural levels.

Some of the basic principles of any architecture should be (given the goal of tracking exponential bandwidths with exponential clock rate improvements):

- No DRAM in the fast path. DRAM Row-Access Strobe times keep it several orders of magnitude slower than required for an active interconnect. Ideas such as MicroUnity's MediaChannel(TM) might help. Perhaps we should call this a "ROMP" (Register-Only Media Processor).

- No statistical behavior. That is, no caches. With small instruction budgets, things must be tightly timed and eschewing any slop in timing (particularly for synchronous traffic) is least risky. Arguments, most importantly the significantly increased flexibility more storage buys, might stimulate use of an L1 cache on-chip. If one were very careful, a case could be made for a very fast multiported SRAM-based L2 cache, but it is preferable to be cautious, lacking measurements and experiments.

- On-chip optoelectronic solutions are an attractive way to get to the logical register file fast enough. It is not completely clear that such things are possible, but if they are, it makes a whole bunch of other things nicer; a single fiber interconnect could easily replace lots of pins.

The basis of this belief is simple arithmetic that engineering may make relevant. Consider the very core of a modern *ca.* \$3000 personal computer, *e.g.*, a 533 Mhz DEC Alpha. The processor can perform $5.33 * 10^8$ register-register operations of 64 bit width in a second. An OC-192 SONET line [PD96] of 9.6 Gb/s can perform the equivalent of $1.5 * 10^8$ moves of 64 bit data per second.

This suggests that we could think about obtaining 3-10 instructions on each word as it passes through a processor. Special-purpose chips (think of functional units in a pipeline) should be able to go even faster. Thus we can imagine an active interconnect as shown in Figure 3.

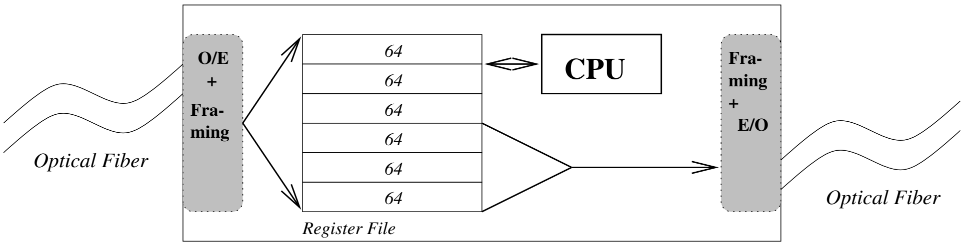

Figure 3: Rough outline of an embedded CPU with fiber/register interconnection logic

Such an interconnect could be resident on switch port controllers, or even more heretically, in the links or in gangs in the switch core. Sequences of such interconnects could be ganged together to achieve more complex functionality. Read-only registers could be tied to values such as queue lengths which are relevant to applications such as self-routing datagrams. There are architectural inspirations for such active interconnects in processor pipelines, in ATM switches with intelligent port controllers (such as Fairisle and Sunshine [Mar96]), and in media processor [Han96] architectures.

3.1 Applications

Here are some applications of a small-instruction budget architecture operating at high speed:

- **Mini-firewalls:**

  ```
  if( R_IN & (INVALID_DST_BITS | INVALID_SRC_BITS) )
  while (!(END_OF_PACKET))
  R_BUCKET <- R_IN;
  else
  while (!(END_OF_PACKET))
  R_OUT <- R_IN;
  ```

  If the framer illustrated in Figure 3 delivered IP packet headers aligned at `R_IN`, a single compare against a mask would decide whether to forward the packet or not; forwarding or dropping could be done in tight loops.

- **Billing Packets:**

  If the idea of Sincoskie for self-paying information transport [SFG+96] is implemented, an arithmetic resource associated with the packet (think of a generalized TTL, for more general resources than a hop count) can be decremented; if it hits zero, the packet is dropped. This can be done before ingress of the packet to any other resources such as a shared switch fabric.

- **Event Counting:** It is very easy for this architecture to count bits, bytes or packets and be queried as packets pass through. This is useful in network measurement and hence management of the network as a system. It may also be useful as part of “flow detection”.

4 Conclusions

The design space for active networking is much larger than often perceived. In particular, if limited flexibility is acceptable, functionality can be injected into the transport path, rather than avoiding the transport path with a control-plane only restriction. This functionality can be specialized hardware, or even a more general purpose CPU. In the latter case, we have speculated about an ideal architecture based on general-purpose CPUs, and shown some simple applications which can work within its constraints.

5 Acknowledgements

Brendan Traw helped with some technical questions, and Jonathan Shapiro improved the paper with his comments.

References

[AAKS98] D. S. Alexander, W. A. Arbaugh, A. D. Keromytis, and J. M. Smith. A Secure Active Network Environment Architecture: Realization in SwitchWare. *IEEE Network Magazine, special issue on Active and Programmable Networks*, 1998.

[ASNS97] D. S. Alexander, M. Shaw, S. Nettles, and J. Smith. Active Bridging. In *Proc. 1997 ACM SIGCOMM Conference*, 1997.

[CJRS89] David D. Clark, Van Jacobson, John Romkey, and Howard Salwen. An Analysis of TCP Processing Overhead. *IEEE Communications Magazine*, 27(6):23–29, June 1989.

[FMS+98] D. C. Feldmeier, A. J. McAuley, J. M. Smith, D. Bakin, W. S. Marcus, and T. M. Raleigh. Protocol boosters. *IEEE JSAC Special Issue on Protocol Architectures for the 21st Century*, 1998. To appear; earlier version available as U. Penn CIS TR MS-CIS-96-34.

[Han96] Craig Hansen. MicroUnity’s MediaProcessor Architecture. *IEEE Micro*, pages 34–41, August 1996.

[HKM+] Michael Hicks, Pankaj Kakkar, Jonathan T. Moore, Carl A. Gunter, and Scott Nettles. Plan: A programming language for active networks. Submitted to PLDI’98. Available in `www.cis.upenn.edu/~switchware/papers/plan.ps`.

[HR98] D. A. Halls and S. G. Rooney. Controlling the Tempest: Adaptive Management in Advanced ATM Control Architectures. *IEEE Journal on Selected Areas in Communication (Special Issue on Protocol Architectures for 21st Century Applications)*, 16(3), April 1998.

[HS97] Ilija Hadžić and Jonathan M. Smith. P4: A platform for FPGA implementation of Protocol Boosters. In *Field Programmable Logic 1997*, number 1304 in Lecture Notes in Computer Science, pages 438–447. Springer-Verlag, 1997.

[Mar96] W. S. Marcus. An architecture for qos analysis and experimentation. *ACM/IEEE Transactions on Networking*, pages 597–603, August 1996.

[NML98] Peter Newman, Greg Minshall, and Thomas Lyon. IP Switching – ATM Under IP. *IEEE/ACM Transactions on Networking*, pages 117–129, April 1998.

[PCB+98] C. Partridge, P. Carvey, E. Burgess, I. Castineyra, T. Clarke, L. Graham, M. Hathaway, P. Herman, A. King, S. Kohalmi, T. Ma, J. Mcallen, T. Mendez, W. Milliken, R. Pettyjohn, J. Rokosz, J. Seeger, M. Sollins, S. Storch, B. Tober, G. Troxel, D. Waitzman, and S. Winterble. A 50-Gb/s IP router. *IEEE/ACM Transactions on Networking*, 6(3):237–248, June 1998.

[PD96] Larry L. Peterson and Bruce S. Davie. *Computer Networks: A Systems Approach*. Morgan Kaufmann Publishers, Inc., 1996.

[PLA] *PLAN documentation on the Web*. http://www.cis.upenn.edu/ switchware/PLAN.

[SAM+98] J. Smith, D. S. Alexander, W. Marcus, M. Segal, and W. D. Sincoskie. Towards an active internet, 1998.

[SFG+96] J. M. Smith, D. J. Farber, C. A. Gunter, S. M. Nettles, D. C. Feldmeier, and W. D. Sincoskie. Switchware: Accelerating network evolution (white paper). Technical report, University of Pennsylvania, URL: http//www.cis.upenn.edu/ jms/white-paper.ps, June 1996.

[TSS+97] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden. "a survey of active network research. *IEEE Communications*, 35(1):80–86, January 1997.

[vdML98] J. E. van der Merwe and I. M. Leslie. Service-Specific Control Architectures for ATM. *IEEE Journal on Selected Areas in Communication (Special Issue on Protocol Architectures for 21st Century Applications)*, 16(3), April 1998.

[WGT98] David J. Wetherall, John Guttag, and David L. Tennenhouse. Ants: A toolkit for building and dynamically deploying network protocols. In *Proceedings, IEEE OpenArch 98*, 1998.