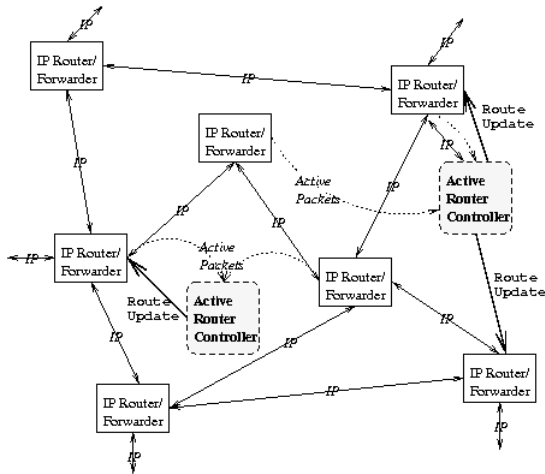


## Active Management of Paths (AMP)

Jonathan M. Smith, CIS Department  
University of Pennsylvania

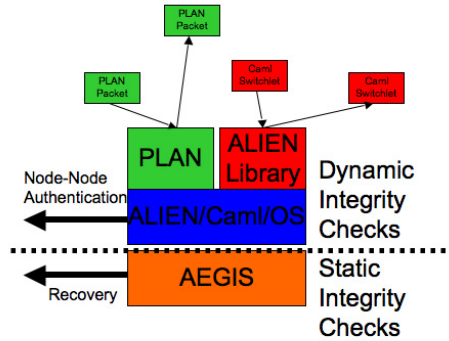


**Figure 1** ARCs in an Internet

Separation of concerns splits routers into technologies for *moving* packets and technologies for *deciding* where they will move. The position taken here is that IP forwarding technology is evolving steadily as a function of line rate and switching improvements, while the control elements have been evolving much more slowly. Active network techniques lacked community pull a decade ago: the Internet was less mature, there was money to be made on maturing it, and its problems were less pressing. Now that problems such as DDoS and business issues such as network neutrality have arisen, active networks are again seen as a solution. For example, NSF's proposed GENI testbed is an active network infrastructure in the programmable switch style of active networking[1], albeit with a focus on sharing ("virtualization") of diverse capabilities amongst sets of nodes, rather than on the nature of the capabilities.

AMP focuses on applying programmability. The AMP network architecture embeds one or more active router controllers (ARCs) in a network. ARCs control the network graph in their vicinity and communicate with other ARCs using digitally signed active packets. The code might be in languages such as OCaml, Packet Language for Active Networks

(PLAN), Java or even declarative languages such as Datalog.



**Figure 2: SwitchWare Node Architecture**

For concreteness, consider PLAN. PLAN is a very high level language in which packet programs are written. Other than some very basic primitives for sending packets from node to node, PLAN's capabilities are defined by extensions to the nodes written in an extension language (PLAN uses OCaml). PLAN has been used to define an Active Internet[2] (by serving as a programmable interoperability layer interconnecting switched Ethernet and IP used as a link layer), and applied to congestion control by discovering the least loaded path to route sequences of packets along. The ability to execute code in the network provides far more transparency to network state than inferences drawn from phenomena such as packet loss or hop count expiry (as with `traceroute`). While its simplicity provides inherent safety, integrity is assured with cryptographic mechanisms in the Secure PLAN system. We thus have an existence proof for a safe, secure method of active node programming, which can be employed for ARCs. ARCs can be used to manipulate conventional routers, Ethernet switches or other forwarding devices. For example, ARCs can directly add and delete routes, extract router states and flow state, and configure interactions with the ARC. More importantly, multiple ARCs can act in concert to manage larger groups of routers. A useful application of this capability is the management of paths, which we call AMP, to achieve capabilities such as dispersity

routing[3], network striping[4], VLANs[5], etc.

### Discovering Paths

One challenge is to discover the topology of the network. If ARCs control routing for a set of pure forwarders, they can use one of a variety of algorithms that have been validated at scale in the Internet. Assuming ARCs are privileged in areas where they are deployed, they can query nearby routers and construct a local graph, which in turn can be used to construct a larger pathmap, as needed. InterARC paths are discoverable with a path discovery protocol such as that used by PLANet to route around congestion[2].

### Exploiting Paths

Conventional routers select paths based on utility estimates, typically using an easily computable metric such as hop count. ARCs can construct independent paths optimized for a variety of criteria including throughput, delay, reliability and security. Multiple paths can be “bonded” together to provide an aggregate with desirable properties. For example, to achieve higher reliability, a protocol can be constructed which uses two independent paths to send packets. Using a timer and an acknowledgement protocol, a new packet can be sent along a path whenever an acknowledgment is received for a packet sent along that path. The probability of failure  $F_{AB}$  for the aggregate path constructed from paths A and B (assuming independence) can be calculated as  $F_A * F_B$ .

Using Maxemchuk’s idea[3] of splitting a message into multiple portions, each of which travels along a separate path, parallelism can be exploited for lower latency. High throughput under load can be achieved by exploiting whatever parallelism is present in the network, using a variety of “striping” algorithms[4]. In addition to the security improvement derived from increased reliability, exploiting multiple paths can also make monitoring by an adversary considerably more difficult.

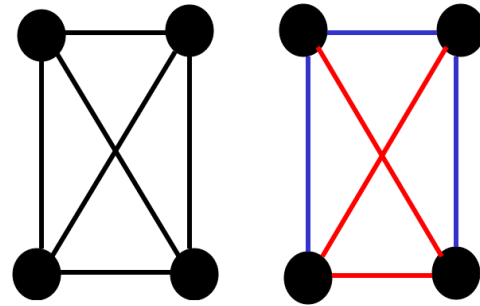


Figure 3: Isolated Sub-graphs

### Isolation

An additional property that may be desired in some circumstances (e.g., for hard real time guarantees, or security “air-gapping”) is isolation – the colored paths illustrate how multiple isolated spanning trees[5] can be constructed where the subnet topology is sufficiently rich in paths.

### References

- [1] D. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. Wetherall and G. Minden, “A Survey of Active Network Research”, IEEE Communications, January 1997.
- [2] M. W. Hicks, J. T. Moore, D. S. Alexander, C. A. Gunter and S. M. Nettles, “PLANet: An Active Internetwork”, Proc. IEEE Infocom, 1999.
- [3] N. Maxemchuk, “Dispersity Routing”, Ph.D. Thesis, University of Pennsylvania, 1975.
- [4] C. Brendan S. Traw and Jonathan M. Smith, “Striping in the Network Subsystem”, IEEE Network, 1995.
- [5] W. D. Sincoskie and C. J. Cotton, “Extended Bridge Algorithms for Large Networks”, IEEE Network 2(1), January 1988, pp. 16-24.