Active Networks

Arturo Azcorra,
U. Carlos III de Madrid, Spain
<azcorra@it.uc3m.es>

Much effort has been placed on developing a packet (cell) switched network technology that provides a continuous range of guaranteed transfer rates (not only a constant scalar rate, but also variable statistical rate), while guaranteeing at the same time a low delay and delay jitter. Frame Relay, ATM and more recently Integrated/Differentiated Services Internet are examples of the interest of the global networking industry in guaranteeing QoS within packet networks. However, much lesser effort has been placed in developing a networking technology that provides a broad range of *functional* services.

The limitations of current networking technologies to allow fast development and deployment of new network services were identified in DARPA along 1995 and have led to a recent proposal of a new network model called Active Networks. In this model the nodes provide an execution environment over which the code used to process a packet is executed. The code may be included in the packet itself or pre-loaded as a result of a previous packet of the same class. The objective is a network technology that allows the fast design and deployment of new network services without requiring the modification of the network nodes.

The key elements of the different Active Network approaches are: the Network API, the Code Distribution Technique and Security techniques. The network API is a virtual (*routing*) machine that interprets a specific language. The expressive power of the language is balanced against the safety of the operations performed by code injected by the users. The current alternatives range from very simple languages, that barely allow the user to select parameter values over a fixed set of operations, to complete languages Turing equivalent. Some reasonable balanced proposals call for powerful languages, but with restrictions that simplify the safety analysis on the code such as prohibiting the usage of loops and pointers. Besides the expressiveness of the language, another issue is the possibility, or not, of storing state information in the Active Nodes. Again, permanent (or soft-state) storage in the nodes allows the development of a broad range of services, but implies that some technique must be provided to perform a fair and optimal allocation of resources among the users. The last aspect related to the network API is its support for the composition of basic-services, referred to as “components”. A flexible and simple support for the dynamic combination of components to easily produce richer network services is considered one of the most important aspects for the success of a network API.

In respect to Code Distribution, there are two main approaches. The first one is the *discrete* or *out-of-ba*nd approach, and the second is the *integrated* or *in-band* approach. Under the discrete approach the code is loaded on the nodes under the responsibility of the network administrator. This has the advantage of off-line control on the safety of the programs, but a big drawback is that it concentrates all the flexibility on the network provider hands. Under the integrated approach, each packet injected by the user “carries” the code needed to process it. This can be accomplished in two different ways: embedded-code or demand-loading. The embedded-code technique implies that the code is really within the packet, and therefore is appropriate for "rare" packets that require simple but specific processing. In the demand-loading technique the code is automatically loaded in the node when the first packet of a given "class" is received. The code remains then within the node in order to process all subsequent packets that belong to the same "class". The usual way to load the code is by requesting it to the node from which the packet was received, inasmuch as the packet has been forwarded by it and therefore this node should have previously loaded the code. Embedded-code and demand-loading can be combined in a single Active Network framework, as they are more complementary than alternative techniques.

The Security area is the one considered less developed on the AN field. Security implies first that only authorized users access the network, each with its corresponding level of resources, and solutions are based on relatively conventional technology such as digital signatures and access lists. Second, it is also required that authorized users, either maliciously or not, do not cause malfunctions to other users or to the complete network. This aspect is typically denoted by *safety* or *robustness*, and requires the development of new paradigms. Distributed resource measurement, fair resource allocators, and code verifiers are some of the algorithms and techniques still to be developed.

DARPA is currently funding several projects, which involve various Universities and companies such as MIT, UCLA, Univ. of Columbia, Univ. of Pennsylvania, BBN, TASC or TIS. These projects are focusing in different aspects of active networks technology such as: execution environments, active node operating systems, specific programming languages, new cryptographic techniques and security issues and potential active applications (i.e. Reliable multicast, active firewalling)
Nowadays, two execution environments seem the most relevant to investigate possible applications of active technologies:

- PLANet, developed at Univ. of Pennsylvania. This active networking testbed is based on PLAN, a programming language especially designed for active networks. PLANet supports both active packets; providing per-packet customizability and active extensions that allows modifying nodes functionality by downloading and dynamically installing Ocaml bytecodes.
  The last version is the 3.2 and was delivered in August of 1999. New security mechanisms have been added in this version, providing access control at authentication time via namespace-adjustment.
- ANTS, developed at MIT. ANTS is a Java based toolkit and provides a node runtime that can participate in a active network and a protocol programming model that allows users to customize the forwarding of their packets. The last version is the 1.2 and was delivered in November 1997. ANTS is the most commonly used due to its simple programming interface and the fact that the language used to program the active network is Java.

The node operating system area is also a very hot topic as showed by the presentation, at the Active Network Demo Workshop sponsored by DARPA and celebrated September 99 at Virginia, of the Janos node OS. Janos is a Java-oriented active network operating system, developed in the University of Utah that intends to provide a solid, resource-aware foundation for future active networks. Janos provides many of the interfaces designed by the active networks Node Operating System Committee, but with a Java binding.

The European Union is also likely to fund several research projects on Active Networks under the IST program. However, as of the date of the production of this document, the projects are technically approved, but still under negotiation and therefore it is not possible to provide formal detail about their content.

References

[Cla88] D. Clark. The design Philosophy of the DARPA Internet Protocols. *Proc. Of ACM SIGCOMM'88*, August 1988.

[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 Comunications Magazine, pp.* 80-86, January 1997.

[TW96] D.L. Tennenhouse and D. Wetherall. Towards an Active Network Architecture. *Proc. Multimedia Computing and Networking 96, MMCN 96,* San Jose, CA, SPIE, January 1996.

[Wet97] D. Wetherall. Developing Network Protocols with the ANTS Toolkit. August 1997.

[WGT98] D. Wetherall, J. Guttag and D.L. Tennenhouse. ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols. Submitted to IEEE OPENARCH'98, San Francisco, CA, April 1998.