Active Networks: an Evolution of the Internet

Giuseppe Di Fatta, Giuseppe Lo Re

CERE-CNR, Viale delle Scienze, 90128

Palermo (Italy)

{difatta, lore}@cere.pa.cnr.it

Abstract

Le Active Networks si propongono quale evoluzione del classico modello di rete a commutrazione di pacchetto. Al tradizionale modello "passivo" di rete basato su una definizione statica del comportamento della infrastruttura di trasmissione dati si contrappone un modello "attivo" in cui i nodi intermedi della rete (switch e router) possono eseguire codice utente contenuto nelle unità elementari di dati trasmessi (pacchetti). Le Active Networks costituiscono un modello di rete "programmabile" nel quale l'infrastruttura di rete viene riconosciuta quale strumento di trasmissione e di calcolo allo stesso tempo e su cui si aprono nuovi scenari di indagine di estremo interesse. Questo articolo da una breve introduzione alle Active Networks, discute i vantaggi che esse introducono e presenta lo stato di avanzamento delle ricerche del settore.

Active Networks can be seen as an evolution of the classical model of packet-switched networks. The traditional and "passive" network model is based on a static definition of the network node behaviour. Active Networks propose an "active" model where the intermediate nodes (switches and routers) can load and execute user code contained in the data units (packets). Active Networks are a programmable network model, where bandwidth and computation are both considered shared network resources. This approach opens up new interesting research fields. This paper gives a short introduction of Active Networks, discusses the advantages they introduce and presents the research advances in this field.

1 Introduction

Active Networks are packet-switched networks in which packets can contain code fragments that are executed on the intermediary nodes [TENN96]. The defining characteristic of an active network is the **capability for users to load software components into network nodes**, dynamically, without explicit reference to any third party. In such a way, Active Networks provide a programmable interface in network nodes to enable the construction of new services.

Traditional computer networks apply the ***store-and-forward*** paradigm. A network consists of a set of nodes interconnected by transmission links. The basic unit of multiplexing of transmission facilities is the packet. Each node performs only the processing necessary to forward packets toward their destination. Nodes receive packets from users or from other nodes, perform a computation based on their internal state and on the control information (header) carried in the packet, and, as a result of that computation,

may forward one or more packets towards other nodes or to users. Networks allow their users to share network bandwidth as a common resource.

Active Networks override this vision by introducing the ***store-compute-and-forward*** concept. Active packets carry not only data but also suitable code portions that will be executed at intermediate nodes as they walk through the network.

Several problems arise from this new approach, the most remarkable of which regards security, but they are balanced by the new capabilities offered by this new technology. Active Networks focus not only on bandwidth but also on other network resources, such as the computing and storage capabilities at end-systems and at intermediate nodes as well. An active network allows its users to write applications that make use of the CPU, main memory and disk space located at the intermediate nodes. It also provides the means to inject user code into these nodes, thus enabling user customisation of network protocols and services.

Active Networks may be regarded as the natural evolution of current ones. Usual packets might be already considered as program carriers, but their functions are so limited that basically they only accept a payload and specify an evaluation destination; moreover the evaluation consists only in the extraction of data. The payload is in fact made of passive data that are just passed to upper services; instead, in an active environment, it is replaced by a program (which may contain data as well). The network designer thus needs to provide the user with a programmable network interface, a virtual machine suitable for the users to inject their active packets into the network. An active environment can thus be divided into a fixed part (the network API) and a variable, user-defined one. If the API defines a complete Turing machine, then the programmer has virtually no limitations in determining the node behavior; on the other hand, the fixed part could accept only some predefined parameters and little variations from a standard behavior are allowed.

In the former case it may be impossible to understand the influence of single nodes on the whole behavior of the net, while the latter one is more or less the common case in the current Internet, where the possible choice for parameters is very limited. Clearly a better choice lies somewhere in the middle.

1.1 Evolution, revolution, or involution

A traditional network architecture is a set of layers and protocols, where each layer offers certain statically pre-defined services to higher layers. In this context Active Networking could be considered a **revolution**. The Active Network [SMITH] differs from traditional architectures primarily in what it does not specify. Instead of defining how the nodes work together to provide the network service, the active network describes functional slots that must be instantiated to provide a particular network service. These slots create a new degree of freedom in network architectures, which, in turn, opens up the opportunity to speed up network evolution and thus accommodate new network types, algorithms and applications. In this way it will be possible to solve one of the most relevant problem of today networks, that is the difficulty of deploying new protocols.

Currently, the standardization process of a new protocol requires times, which are measurable in years. The introduction of a programmable network infrastructure will allow the users to bypass this long and tedious process by identifying new needs that can occur in the network and by programming the network to manage them**.**

On the other side, purists of Networking concepts have even considered Active Networks an **involution**. The hierarchical structure of the layered architecture follows strong principles and allow to cope with the design of complex systems. Active Networking seems to break some fundamental rules (e.g. the end-to-end argument [BHATT1]) and to introduce anarchy in the rigorous organization of the layered structure.

Active Networks research community like to consider it an **evolution** of traditional network architectures "*motivated by both technology push and user pull*" [TENN97].

Active Networks concepts are not totally new. Specific solutions to many problems in networking have been proposed and even adopted, which perform user-driven computation at nodes "within" the network (e.g. firewalls, web caches and proxies, multicast routers, mobile proxies, and video gateways).

Furthermore, a lot of recent trends in Networking could be considered a subset of the Active Network Architecture: for instance VLAN in PHY layer, Multiprotocol Router, RSVP, RTP, Application Layer Routing are all functions that an active network can provide. Active Networks can be used as a common infrastructure to solve both the current problems and the ones will come with no need for many ad-hoc solutions.

The goal for active networking is to build a network infrastructure based on programmable open nodes where it is possible to deploy programs dynamically into node engines. Active Networks research is, namely, developing mechanisms to increase the flexibility and customizability of the network and to accelerate the pace at which software is deployed.

1.2 Two different approaches

There are two possible approaches to build Active Networks: a discrete and an integrated approach.

The **Discrete Approach** is also called **Programmable Node Approach** because programs are injected into the programmable active node separately from the actual data packets that traverses through the network. The "user" would send the program to the network node (switch or router), where it would be stored and later executed when the data arrives at the node, processing that data which are associated to the program. The data have some information that would let the node decide how to handle it or what program to execute.

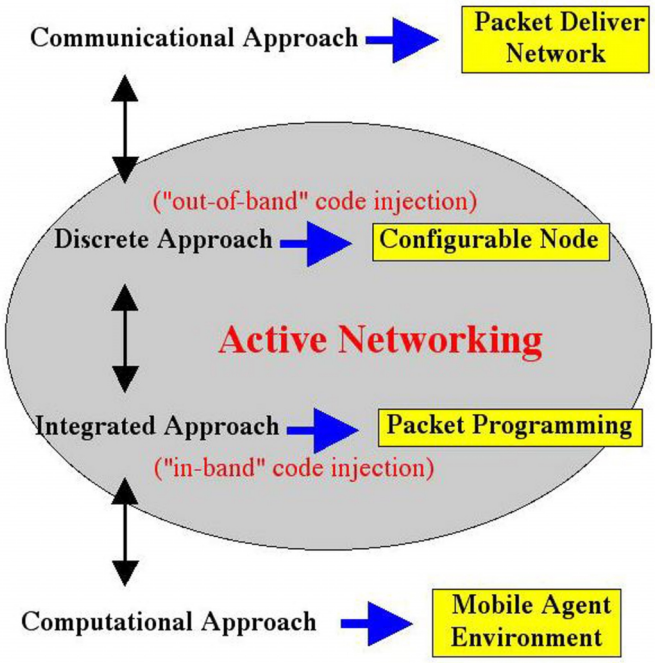

Figure 1: different approaches in proposed architectures

In the **Integrated Approach**, also termed as the **Encapsulation Approach**, the program is integrated into every packet of data, which is called "**capsule**". Each capsule contains a program fragment that may or may not have some embedded data. When such a capsule arrives at the active node, an Execution Environment (EE) interprets the program and takes some action according to the interpretation of the program (like sending the embedded data to the destination. In this approach, each active node has a built-in mechanism to load the encapsulated code, an execution environment to execute the code and a relatively permanent storage where capsules would retrieve or store information.

The following table gives a summary of the main characteristics of the two approaches.

Approach	Node Programmability Level	Programs	Code injection vs data transmission
Discrete	Configurable, extensible node	Switchlets, modules	Out-of-band
Integrated	Execution Environment	Packet programming	In-band

Table 1: Discrete and Integrated Approaches

2 Sciences from which Active Networks inherit

The concept of active networking explicitly emerged [TENN97] from discussions on the future directions of networking systems within the broad DARPA research community in 1994 and 1995. However, the idea of messages carrying procedures and data is a natural step beyond the traditional circuit and packet switching systems, and was already applied in the past to rapidly adapt a network to changing requirements. According to this point of view, Softnet [ZF83] should be considered the earliest (1983) active network. Softnet is a packet radio network developed at the University of Linkoping (Sweden), where each node consists of a dual-processor: the node processor for the system management, and the link processor to control the radio transceiver and the interface to the node processor. Each packet contains a FORTH program to be executed on network nodes. Softnet proposes a distributed network of computational nodes and provides programmability in all layers of the network infrastructure.

Although the difficulties in the Softnet project were related to inefficiency mainly because on the early 1980s software technologies had not yet reached a sufficient development, in the 1990s several areas of research enabled the Active Network paradigm to emerge as a programmable Network architecture. On the other side the telecommunication market has been demanding the introduction of new services and even a common definition of a framework which enables the dynamic and fast introduction of new services.

In this section we briefly discuss areas which can be considered relevant to the development of Active Networks concept.

2.1 Programmability in Telephony

In Telephony infrastructures programmability of electronic switching systems provided service creation and deployment.

During the mid-1980s, regional Bell operating companies (RBOCs) began to request features that met the following objectives: rapid deployment of services in the network, vendor independence and standard interfaces, and opportunities for non–RBOCs to offer services for increased network usage. Bell Communications Research (Bellcore), today Telcordia Technologies, responded to this request and

developed the concept of Intelligent Network [IN]. An Intelligent Network (IN) is a service-independent Telecommunications network. That is, intelligence is taken out of the switch and placed in computer nodes that are distributed throughout the network. This provides the network operator with the means to develop and control services more efficiently. New capabilities can be rapidly introduced into the network. Once introduced, services are easily customized to meet individual customer's needs.

The service-independent Advanced Intelligent Network (AIN) followed IN and demonstrated that programmability can accelerate service introduction in the context of telephone calls.

2.2 Configurable Network Protocols

The philosophy of configurable network protocols, such as *Application Level Framing* (*ALF*) [CLAR90], *x-kernel* [HUTC91] [O'MA92], and *Protocol Boosters* [MALL96] [FMS+98], is a design guideline that includes the semantics of the application in the design of its transport protocol. It argues that the roles of application and network must be matched for efficient processing. The aim of these protocol architectures is to provide a communication environment that satisfies the requirement of a large range of distributed applications.

Application Level Framing is a communication system architecture where applications play an important role in the data transmission process, as they best know the characteristic and the requirements of the information being transmitted. The *ALF* principle states that “information should be packetized by the application into Application Data Unit (ADU’s), each of which should be at the same time a unit of transmission, a unit of control, and a unit of processing” [CKD98].

The *x-kernel* is an object-based framework for implementing network protocols. It defines an interface that protocols use to invoke operations on one another and a collection of libraries for manipulating messages, participant addresses, events, associative memory tables (maps), threads, and so on.

The *x-kernel* has been adopted as a research platform for investigating end-to-end issues related to computer networks, i.e. it focuses on the design and the implementation of network software running on the end systems connected to the network.

The *Protocol Boosters* work shows that programmability can be useful for dynamic adaptability. In that work, they were able to adapt to conditions such as increased loss within the network by dynamically adding error correction functionality around the lossy portion of the network.

2.3 Mobile Agents and Interpreted Programming Languages

There exist several languages for programming server-to-client code migration (e.g. Java [SUN95], Safe-TCL [BOR94], Python [VROS], Omniware [COL95]) and several others are specifically designed for programming mobile agents (Phantom [COU95], Obliq [CAR95], Telescript [TELE95]). A mobile agent carries a transient execution environment along with code; this allows a computation to start at one node in the network and migrate itself to other nodes.

Active packets [CHEN] have an obvious similarity to the better known mobile agents, and serve a complementary purpose in some way. Although there is no universal definition of software agent, generally agents are regarded as software programs designed to carry out a specific function or task on behalf of a user. Both active packets and mobile agents view the network as a distributed programmable environment, but they are different in orientation and operation. Active packets are concerned mostly with the customization of packet handling service related to a user's connection. For this reason, they work within the network on the network itself, by a process of code movement and remote execution.

In contrast, mobile agents may typically have more intelligence and self-directed autonomy to carry out their function, which may be more oriented toward mobile computation, i.e. towards hosts rather than the network infrastructure.

In this way, agents can be viewed as an enabling technology, as they provide many support mechanisms characteristic for AN and enable the implementation of autonomy characteristic.

2.4 Extensible Operating Systems

Several research efforts have been focusing on safe, application-specific extensions to operating systems to provide and manage a flexible and controlled access to local system services [CHD94]. Many of these works, such as *SPIN* (1995), *Exokernel* (1995), *ASHs* (1996), *Plexus* (1996), *Scout* (1995), result in techniques for safely extending a software subsystem.

SPIN [BER95] allows users to download code into the kernel. It adopts Modula-3 which is a strongly typed language with automatic garbage collection. The type checking of the Modula-3 compiler is used to ensure domains of protection at the level of binding names for access to objects. Programs signed by the compiler may be dynamically loaded into the operating system.

Exokernel [ENG95] allows applications to design their own OS abstractions and to extend OS for increased performance or functionality.

ASHs [WAL96] allow user-defined handlers to be run by the kernel in response to packet arrivals.

Plexus [FIB96] allows application-specic communication protocols to be incorporated into the kernel.

Scout [MONT95] [MOSB97] is an operating system implementation based upon *x-kernel* with a special focus on network appliances. It is mainly end-system oriented with a router module which implements QoS functionality and IPv6. It allocates and schedules resources on a “path” basis and applies a number of optimizations intended to increase throughput and decrease latency. Many of the techniques may be applicable to programs loaded into network nodes.

2.5 Distributed Object technology for networking

Distributed object systems [RSW98] have become a foundation technology and operational platform for large, complex communications network. CORBA (Common Object Request Broker Architecture) and DCOM (Distributed Component Object Model) represent the current technologies in this field. An **object** is a piece of software representing any service or information providing entity, such as a document, a network element, a media stream or device, or a communication session. An **object class** represents potential objects sharing the defined characteristics of the class.

A **distributed object system** is a dispersed set of objects that can request services from one another, through a communication mechanism, using interfaces defined in a consistent IDL (Interface Definition Language).

Distributed objects technology offers important benefits to networking systems such as modularity, reusability, scalability, and reliability, which allow to reduce the service deployment cycle.

Nevertheless, these systems present the inconvenient to be completely static, which means they do not allow to dynamically inject new software behaviors into the network, but to simply use predefined static procedures available on the nodes. These constraints are overridden by the Active Networks technology.

3 Potential use of Active Networks

3.1 New applications

There has been a long debate about the "killer application" in Active Networks. The evolution of computer networks towards the active network paradigm strongly depends on the actual benefits that can be obtained by applications. We feel that many of these benefits fall into the following categories.

Availability of Information Held by Intermediate Nodes

Mobile agents can be encapsulated and transported in the active code of application capsules. They can retrieve and extract pieces of information held by intermediate nodes in a more effective way than through remote queries from the application itself. For instance, an agent could make use of active code to look-up the routing tables of an intermediate node and select some entries according to a given criterion. It can either send such extracted information back to the application, or it can use the information to take timely decisions autonomously from the application. More examples can be found in network management issues, such as congestion control, error management, traffic monitoring. A meaningful example is the one related to the customization of the routing function. A mobile agent could be devoted to the evaluation of the path for the application's data flow, according to the user's QoS specification. Each application could set up its own control policy or exploit a common service (the default per-hop forward function).

Data Processing Capability along the Path

Application-specific functions installed into an intermediate node could access and modify transient data addressed to other nodes. Such modification could be due to the current state of the network or to particular receiver needs. Data format translations, different compression levels, document encryption/decryption are some of the examples. Multicast transmission is an instance where the benefits appear more evident. Functions, dynamically deployed in intermediate nodes, can manage the joining of new users, or they can dynamically modify the multicast tree to optimize bandwidth utilization, or, again, adapt the data format to different user specifications.

Audio and video conferencing systems have been proposed in [Baldi], [Banchs], in which agents are located in crucial nodes where the transformation of the streams is needed. Each agent is in charge of the replication of the information for different users. It can also adapt the data flow to different bandwidth requirements and to the network load.

Adoption of Distributed Strategies

Active networks applications can easily implement distributed strategies by spreading application mobile agents in the network. Examples of this new potentiality are given by existing applications such as web proxies [TENN97], stock quotes and on line auction applications [WETH1], distributed firewalls [TENN97], and the distributed management of multicast trees [LI]. A particular application proposed in [VV97] is an ad-hoc mobile firewall, whose aim is to inhibit the annoying denial of service attack known as the SYN-Flooding attack. This attack consists of a big number of pending TCP connection requests to a server thus to waste all its memory resources. To discover the source of the incoming requests and to stop the attack, the defender injects into the network a defense mobile agent that is able to recognize the intruder packets and to stop them in intermediate nodes closer and closer to the attacker's node.

3.2 New network services and resource optimisation

The code injection technique is the key of the flexibility of an active network, and makes the development of new protocols, services, and other network applications straightforward. Tests on new protocols can be quickly performed on the real network, and not just simply simulated. The updating of network device software with complex dependencies can be remotely accomplished. The need of a greater celerity for delivering the software to network devices arises from the knowledge that the difficulty in introducing several Internet enhancement attempts (RSVP, MBone, IPv6) was also due to the impossibility of accomplishing the necessary actions on the network devices. The *diffserv* working group is proposing an architecture [Blake] to support different services other than best-effort forwarding. Per-Hop Behaviors (PHB) are the bricks with which these new services can be built. Per-Hop Behaviors are dynamically allocated in the network nodes and the active network methodology could be very suitable for this aim.

New services that could be easily implemented in an active network are application-driven routing, already discussed, and parallel routing, where several parallel paths substitute the single path of unicast transmission.

4 The main players

4.1 Fora and Consortia

Two schools of thought have emerged as far: OPENSIG and DARPA related projects.

OPENSIG is an international research and industry community, which was formed in 1995 to advance the concept of open signaling and network programmability. Shortly, OPENSIG means programmable networks in a telecommunication-oriented approach.

The work of the OPENSIG group has focused on the definition, implementation and experimentation of OPEN PROGRAMMABLE NETWORKS. Having an impact on products and services of the next generation networks is a key goal of the OPENSIG working group. An important step in this quest was the submission of a Project Authorization Request to IEEE by Columbia University, Ericsson, ISS and NEC with the goal of standardizing "Programming Interfaces for Networks" (PIN). This request was approved on December 8, 1997, as proposed IEEE standard for Application Programming Interfaces for Networks (IEEE P1520).

DARPA AN projects (most of the AN projects) mainly focus on IP routers. Their approach, which means the AN approach, is more dynamic than OPENSIG's semi-static network programming interfaces.

Here, the scenario spaces between two extremes:

-a single packet boots a complete software environment for all packets arriving at the node;

-a single packet (capsule) modifies the behavior of the node seen only by that packet.

In the DARPA projects approach the functionality of the active network node is divided between the Execution Environments (EE) and the Node Operating System (NodeOS). EEs are responsible for implementing the network API, while the NodeOS manages the access to local node resources by EEs.

Each EE is similar to a "shell" program in a general-purpose computing system, providing an interface through which end-to-end network services are provided to users. The architecture allows for multiple EEs to be present on a single active node. All user access to node resources (including the transmission bandwidth) is provided through an EE.

The NodeOS provides the basic functions from which EEs build the abstractions that make up network APIs. It manages the resources of the active node and mediates the demand for those resources, including transmission, computing, and storage.

The Active Network Encapsulation Protocol (ANEP) provides the multiplexing capability to route active packets to a particular EE in a node.

4.2 Proposed standards

Several companies and laboratories produced a IEEE standards development project, **IEEE P1520**. This project envisions tomorrow's telecommunications networks as a giant computer - a fully programmable machine - that delivers advanced voice, data and video services globally.

The proposed standard specifies, in industry-standard Interface Definition Language, a set of programming interfaces for distributed access to switching functionalities by service control entities, including but not restricted to signaling services entities.

The types of switching units considered in this standard include Asynchronous Transfer Mode (ATM) switches, circuit switches running Signaling System N. 7 (SS7) and IP routers.

The Active Network Encapsulation Protocol (**ANEP**) [ANEP] [ANIP6] is an effort towards the interoperability among different EEs. In ANEP terminology, a packet consists of an ANEP header and a payload.

ANEP header includes a Type Identifier field; well-known Type IDs are assigned to specific EEs. (Presently this assignment is handled by the Active Network Assigned Number Autority). If a particular EE is present at a node, packets containing a valid ANEP header with the appropriate Type ID will be routed to the appropriate EE.

In the DARPA program on AN various working groups were formed to discuss the design and development of the components for successfully building active networks. Each working group is developing an architecture document through discussions and contributions from the DARPA active networks research community, and they are available for public release as "ARFCs". The **DARPA Drafts** cover four topics: architectural framework [ARCH], node OS [NODEOS], composable services [CSAN98], and security aspects [SEC98].

5 Active Networks roadmap

The charts in figure 2 show respectively the evolution of some Active Networks related projects, developed since 1994, and the research environments from which they have been originated.

Despite the popularity of AN idea, it has not taken hold in general. In [MN00] authors claim that this is because current active packet systems are not sufficient practical.

Current active network systems try to represent a tradeoff between the three fundamental aspects represented by safety, efficiency, and flexibility. Increased safety may require restrictions on flexibility or have efficiency overheads, while added flexibility may introduce safety holes or require extra computation that decreases efficiency. If we setup a three-dimensional space using these three axes, we can plot active packet systems in this space.

Referring to the measure of practicality as the “volume” in the design space as in Figure 2, it is common opinion that it is possible to create a second-generation active packet system which is more practical than IP. The main tactic will be to make gains in flexibility without sacrificing safety or too much efficiency.

In the last period a great emphasis has been put on the concept of Application Layer Active Networking (ALAN). ALAN extends the concept of the dynamic software loading into the service layer. In order to accommodate this extension, the design of an active node will have to cover a broader range of activity than the conventional active node architecture as developed in the DARPA AN projects. In such a context a distinction is made between two flavors of "activeness", i.e. active routers and active servers. An active router is a flexible version of a conventional network router, i.e. it supports active routing operations, but it does not support application layer processes. The flexibility that can be permitted in an active router is therefore tightly constrained.

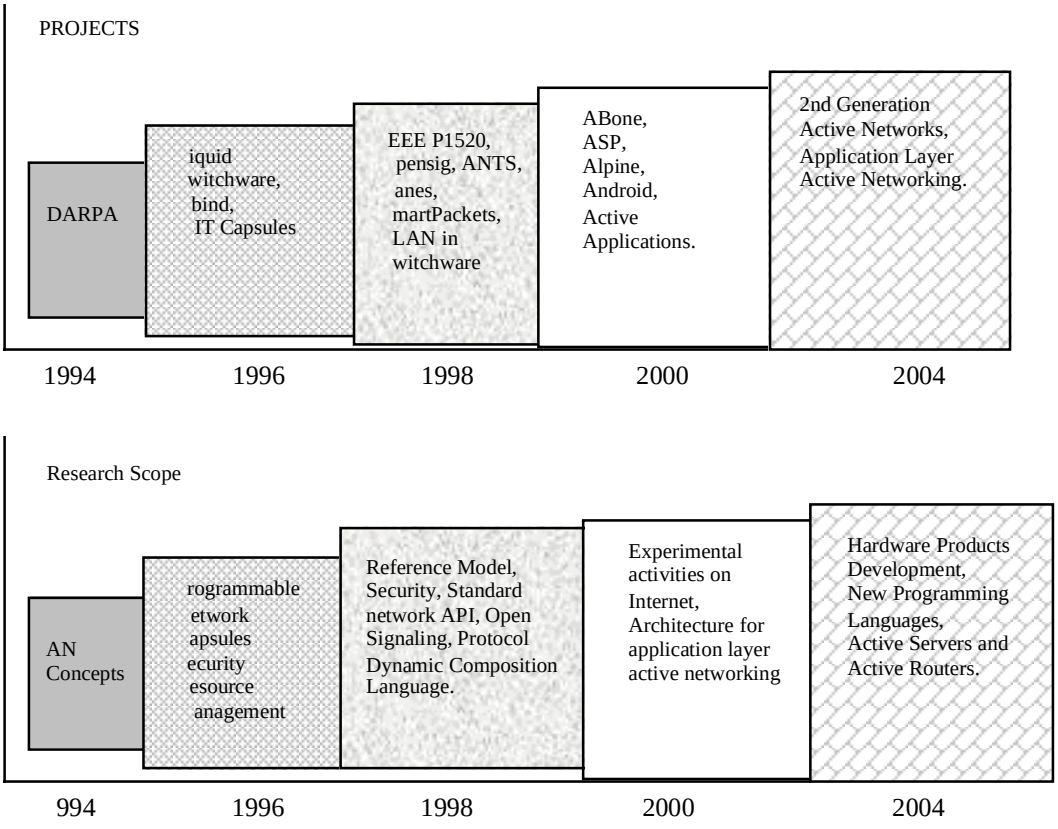

Figure 2 – active networks roadmap

An active server is the second type of device that can offer dynamic programmability. It runs a full protocol stack and for this reason it is logically an end system, although it may be physically associated with router and serve as network intermediary from the user perspective. In this way the control that can be delegated to general users can be much greater than in an active router. This vision solves many of the problems related to security issues in the traditional approach of active network and allows a greatest degree of service programmability for the users.

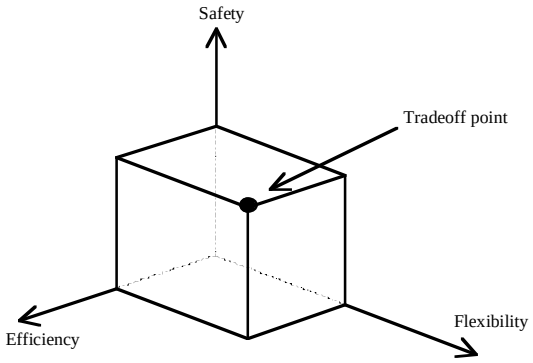

Figure 3 – Active system design space

Looking ahead to next future we can imagine a scenario where it is possible to observe:

- the development of a standard reference model,
- a greater involvement of industries,
- the distinction between active routers and active servers,
- the development of faster platforms capable of processing the active packets,
- the extension of the paradigm to mobile and ubiquitous devices,
- the development of more suitable languages and software platforms.

References

[ANEP] D. Alexander et al, Active Network Encapsulation Protocol. Draft, July 1997. Available at http://www.cis.upenn.edu/ switchware/ANEP/.

[ARCH] Architectural Framework for Active Networks Active Network Work. Group, Draft, July 27, 99

[BALDI] Baldi, M., Picco, G., Rizzo, F.: Designing a Videoconference System for Active Networks.Proceedings of the 2nd International Workshop on Mobile Agents, Stuttgart, September 1998

[BANCHS] Banchs, A., Effelsberg, W., Tschudin, C., Turau V.: Multicasting Multimedia Streams with Active Networks. Technical Report TR-97-050, International Computer Science Institute, Berkeley CA

[BER95] B. Bershad et al. "Extensibility, Safety and Performance in the SPIN Operating System", In Proceedings of the 15th ACM Symposium on Operating System Principles (SOSP-15), pages 267–28, Copper Mountain, CO, 1996.

[BHATT1] Bhattacharjee, S., Calvert, K.L., Zegura, E.W.: Active Networking and End-to-End Arguments. IEEE Network Special Issue on Active and Controllable Networks, vol. 12, n. 3, May-June 1998

[BHATT2] Bhattacharjee, S., Calvert, K.L., Zegura, E.W.: On Active Networking and Congestion. Technical Report GIT-CC-96-02, College of Computing, Georgia Tech.

[BOR94] N. Borenstein. “E-Mail With a Mind of Its Own: The Safe-TCL Language for Enabled Mail”, Proceedings of IFIP International Conference, Barcelona, Spain, 1994.

[CAR95] L. Cardelli, “A Language with Distributed Scope”, In Proceedings of ACM Principles of Programming Languages, 1995.

[CHEN] Chen, T. M.,: Evolution to the Programmable Internet. IEEE Communications Magazine, vol.38, n. 3, March 2000, 124 - 128

[CHD94] D. R. Cheriton and K. J. Duda, "A caching model of operating system functionality", In Proc. of the First Symposium on Operating Systems Design and Implementation, 1994.

[CKD98] I. Chrisment, D. Kaplan, and C. Diot, “An ALF Communication Architecture: Design and Automated Implementation”, IEEE JSAC, Vol.16, N.3 (pagg.332-344), April 1998.

[CLAR90] D. D. Clark and D. L. Tennenhouse, "Architectural Considerations for a New Generation of Protocols", In SIGCOMM '90, 1990.

[COL95] Colusa Software, “Omniware: A Universal Substrate for Mobile Code”, White Paper, 1995.

[COU95] A. Courtney, ”Phantom: An Interpreted Language for Distributed Programming” In Proceedings of USENIX Conference on Object-Oriented Technologies, 1995.

[DARPA] http://www.darpa.mil/ito/research/anets/

[ENG95] D. R. Engler et al. "Exokernel: An Operating System Architecture for Application-Level Resource Management", In 15th Symp. on Operating Systems Principles, 1995.

[FIB96] M. E. Fiuczynski and B. N. Bershad, "An extensible protocol architecture for application-specic networking", In Proceedings of the 1996 Winter USENIX Conference, 1996.

[IN] Telcordia Technologies, Inc.Intelligent Network (IN) Tutorial http://www.iec.org/tutorials/in/

[LI] Li-wei, H.L., Garland, S.J., Tennenhouse, D.L.: Active Reliable Multicast. IEEE INFOCOM'98 San Francisco, USA 1998

[MOSB97] David Mosberger. "Scout: A Path-based Operating System", PhD Dissertation, Department of Computer Science, University of Arizona (July 1997)

[MONT95] A. B. Montz, D. Mosberger, S. W. O'Malley, L. L. Peterson, T. A. Proebsting. "Scout: A Communications-Oriented Operating System" Hot OS (May 1995).

[MN00] J.T. Moore, S. M. Nettles, “Towards Practical Programmable Packets”

[NODEOS] NodeOS Interface Specification AN Node OS Working Group Draft, January 24, 2000

[SMITH] Smith, J. M., Calvert, K.L., Murphy, S. L., Orman, H. K., Peterson, L.L.: Activating Networks: A Progress Report. IEEE Computer, Vol. 32 N. 4, April 1999, 32 - 41

[SUN95] Sun Microsystems Inc. “The Java(tm) Language Environment: A White Paper”, 1995.

[TELE95] J. E. White, “Telescript Technology: Mobile Agents”White Paper, 1995.

[TENN97] Tennenhouse, D. L., Smith, J.M., Sincoskie, W.D., Wetherall D.J., Minde, G.J.: A Survey of Active Network Research. IEEE Communications Magazine, Vol. 35, No. 1, January 1997, 8

[TENN96] Tennenhouse, D. L., Wetherall, D.J.: Towards an Active Network Architecture, Computer Communication Review, Vol. 26, No. 2, April 1996

[VV97] Van C. Van, "A Defense Against Address Spoofing Using Active Networks", Master's thesis, Department of Electrical Engineering and Computer Science, MIT, May 1997.

[VROS] G. Van Rossum“Python Tutorial”

[WAL96] D. A. Wallach et al. "ASHs: Application-specic handlers for high-performance messaging", In SIGCOMM'96. ACM, 1996.

[WETH1] Wetherall, D.J., Legedza, U., Guttag, J.: Introducing New Internet Services: Why and How. IEEE Network Magazine Special Issue on Active and Programmable Networks, vol. 12, n.3, May-June 1998

[WETH2] Wetherall, D.J., Guttag, J., Tennenhouse, D.L.: ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols. IEEE OPENARCH'98, San Francisco, CA, April 1998

[ZF83] Jens Zander, Robert Forchheimer, "SOFTNET - An approach to high level packet communication", Tech. Rep., Department of Electrical Engineering, Linkoping University, 1983.