A Framework for Developing Mobile Network Services*

M. Sifalakis†, S. Schmid‡, T. Chart†, A.C. Scott†

† Computing Department, Lancaster University,
Lancaster, LA1 4YR, U.K.
{mjs, chart, acs}@comp.lancs.ac.uk

‡ NEC Europe Ltd, Network Laboratories,
69115 Heidelberg, Germany
schmid@netlab.netlab.nec.de

Abstract. Service mobility is a highly desirable feature of future networks and will be a key factor in supporting mobile users and meeting the increasing demand for service resiliency and efficient resource management. Currently the main approaches for developing mobile services are active capsules and mobile agents but these are limited by design and fail to deliver sufficiently generic and broadly scoped solutions. In this paper we propose a framework for developing mobile (active) network services that aim to provide a more generic functionality over that offered by current approaches, promoting a more flexible and intuitive way of developing mobile applications.

1 Introduction

Despite these manifold developments of active and programmable network research, little attention has been paid to improving aspects of service mobility. So far, the main model considered for migration of code and state of a running service from one active node to another is that of active capsules [1, 2, 3], whilst most programmable network solutions [4, 11, 20] account only for out-of-band loading of code rather than migration of executing 'active services'. However, active capsule based approaches are quite restrictive in terms of code and state mobility as they operate in-band only and at a packet level granularity (i.e. carrying code and state in a single data packet). This tends to restrict the implementations to very simple and short-lived services, as both state and code for the service have to fit in a single data packet and code only executes while the packet is forwarded through a node along the data path. Furthermore, service mobility is restricted to the routing path of the packet hosting the service.

Service mobility is expected to play an important role in future network environments for various reasons. Mobile services can improve the user experience as he/she roams between heterogeneous and diverse network environments. Another key aspect of service mobility is its application for efficient resource management usually by means of load sharing, or by protecting network resources from being exhausted. Last but not least service mobility can facilitate service resiliency as it enables service-and-

* This work is part of the research funded by the EPSRC grand GR/R31461/01 in Lancaster University

state migration to network locations where resources and network availability are plentiful.

In developing generic and flexible solutions to tackle these classes of problem one needs to address a multiplicity of issues such as the handover of network flows, the composition of autonomic services through cooperating control components, the management of active services outside the data path, and so forth. Technologies such as mobile agents and active capsules provide ways of tackling only a subset of the problem. Yet, as both of these technologies were introduced with a specific application domain in mind, they are either not scalable or lack the features and capabilities that would make them more generic and applicable to a wider range of applications.

In this paper we propose a framework for developing active services that can be mobile and migrate to different programmable nodes at run time. The novelty lies on the fact that it enables code and state migration in an application independent way by mitigating the functionality at the execution environment (EE). This type of functionality was not previously considered in programmable network based solutions and was therefore expected to be internal in the applications that need it. To provide a generic model that is flexible, as well as effective, we tried to incorporate the benefits of active capsule solutions and out-of-band mobile agent based technologies.

The remainder of this paper is organised as follows: In the next section, we discuss the motivation for the work by considering a scenario that aims to illustrate the limitations of existing technologies, namely mobile agent and active capsule based solutions. In section 2, we investigate the requirements of a framework supporting the development of mobile services. Then, in section 3, founded on the requirements of section 2 we present the proposed architecture. In section 4 we exemplify the intended use of the framework with an example responding to distributed denial of service attacks. In section 5 we consider related work in the area of service mobility and the tools to facilitate it. Finally in section 6 we conclude this paper summarising our work.

1.1 Motivation

Before we examine the requirements of a framework for mobile active services, we consider a scenario (Figure 1) that exemplifies the problem domain at hand. Through this example we try to expose the limitations and inefficiencies of other approaches such as mobile agents or active capsules to facilitate generic service mobility. Scenarios such as the one described here motivated the development of the proposed framework.

In figure 1 nodes B, C and D are active routers, node A is an end node featuring active network support and node E is a commodity network node. Node A wants to exchange traffic with node E using a service deployed on the active routing infrastructure provided by the other three nodes (B,C,D). Initially A sends an active service request to B, which is the first active node on its data path towards E. Node B, intercepts the request and installs the requested active service. At the same time node B forces the installation of a flow monitoring and management active service component on node A and node D. Finally the service provisioning is activated and data is routed from A through B and D to E. At a later time due to some external factor (for example, congestion, resource utilisation, etc) node B cannot continue providing the active

service for node A. Instead of shutting down the service, node B may force the migration of the active service to another active router (node C); subject to resource availability. Using the flow management components previously loaded onto nodes A and D the data flow can be redirected through node C so as to continue providing an uninterrupted service. Alternatively in an even more "extreme" case where node B dies suddenly, the management components on nodes A and D could detect the "malfunction" and correct the problem by re-installing the service on node C and recover if possible the flow state.

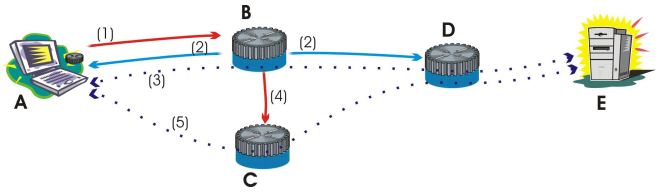

Figure 1. Service Mobility

Despite the seemingly specialised setup of this example, it is by no means restrictive and rather broadly scoped. We have tried to include in it all the characteristics that need addressing when targeting the generic provisioning of mobile services: the support for in-band and out-of-band programmability, the support for service mobility in the data, management and control planes, the support for service composition and integration, and the need to support user space services as well as low-level network services (deployable in the forwarding path).

Using active capsules for the aforementioned scenario could address some of the issues but not all (at least not without highly specialised design or extensions to the basic concept). For example, it would be easy for an active capsule based solution to trigger the installation of the active service on node B in the first place, since it resides on the data path. However, using capsules to control the redirection of data flows is counterintuitive. For this, the active node needs information about alternative nodes that are ready and able to take over, necessitating long-lived state to be gathered and managed by the service (especially for the situation where node B and C are more than 2 hops apart). Unfortunately, active capsule based services are typically limited as the service state migrates with the capsule packet – few systems allow inter-capsule state to be stored at intermediate nodes and this is always restricted by the transmission path.

Also, as active capsules target in-band programmability, and only along the data path, another limitation is the lack of support for the out-of-band transfer of service state required for migrating the service from node B to C. Finally, as active capsules advocate a per-packet processing approach, they do not scale and introduce unnecessary overhead when coarser grained processing is preferable (e.g. per flow).

In addition to providing support for processing active services, there is also need for a management framework that can control service components running on active nodes with regard to node local state (for example, during the migration or "cloning" of a service in response to resource extinction). Although the use of mobile agent technologies may seem like a solution, their use would only introduce additional prob-

lems as they are typically implemented as high-level user-space solutions and are unsuitable for low-level data processing on the forwarding path. Aside from the obvious performance implications, this severely limits the range of applications that can be deployed using mobile agents. Returning to the example scenario in figure 1, if the active service provided by node B was a media caching or a NAT component, it would be impractical to implement it using conventional mobile agent systems.

As a result, to overcome the aforementioned shortcomings of one or the other technology, we anticipate the need for a framework for developing mobile network services that, in addition to being flexible, scalable, extensible and generic, combines the strengths of existing mobile agent and active capsule technologies. The framework proposed in this paper is based on the LARA++ [4] component-based active router architecture. The extremely flexible design of LARA++ enables functionality in the data path as well as the control path, allows deployment of mobile services vertically in the network stack and accounts for micro- as well as macro- composition[†] (see section 2.1 for details).

The benefits of introducing a framework for the development of mobile active services as opposed to promoting the direct top-down design of active applications as mobile services are twofold since it entails advantages both for the active node administrators and for the service developer:

- It promotes a single point of trust for the active node administrator. If the framework is trusted by the active node policies, then so are the applications running within it.
- If the framework provides a reasonable level of security, it removes the need for the developer to build application specific protection mechanisms
- It simplifies the development of mobile active service components through design reuse
- It introduces a level of abstraction that reduces apparent complexity and improves the manageability of the system

2 Framework Requirements Specification

In developing a framework to support mobile active services, one must consider the range of supported functionalities. The overall design is dictated by a set of fundamental principles that advocate its viability in a real world environment and its usability by future applications. Before we describe our framework, we discuss the fundamental requirements and basic design principles that have guided our design:

- *Portability* over any existing or emerging active node platform. This will be typically guaranteed by means of the execution environment (EE) that hosts the framework.
- *Extensibility* without re-engineering the system or violating the specification.
- *Isolation* of individual active services in order to prevent unwanted interaction between services executing in the same EE.

[†] We define Micro-composition as being within the framework and Macro-composition as being outside the framework on the active node or network.

- *Modularity* to facilitate dynamic binding to the framework at run-time. Individual components of the service components at run time.
- *NodeOS and EE compatibility*.

2.1 Framework Functionalities

The main functionality of the framework should be based on a mechanism that enables code migration in a (semi-) transparent way to the mobile services. Although the framework services should be ubiquitously provided to the mobile service, yet it must also be possible to control/customize them whenever needed.

An important requirement is the support of an open and flexible mobility model facilitating both, *forced mobility* as well as *intended mobility*. The former refers to the case where the mobile active service is 'forced' to move by the framework as a direct result of some external. Forced mobility will typically be as a result of decisions intended to ensure service resilience and will account for node-related problems such as resource exhaustion and network related issues such as routing problems. In the case of 'intended mobility', the active application itself decides to move in response to application specific conditions.

So far we have argued that the most important reasons for enabling service mobility are the support of mobile users and the provision of service resiliency. These two reasons are motivated from two different application domains. Derived from this distinction also comes the need distinguish between the active nodes that can host the service in case of forced migration (*alternative active nodes*), and those that can be chosen by the application as candidates to host the mobile service in case of intended migration (*next active nodes*).

Once an event necessitating service migration occurs, different way of migrating services can be chosen depending on the situation. For example, if the service code is not broadly available on the new active node as might be the case with a personalised user service, it might be necessary to move both the code of the active service and its state. In other cases, where the service component is already cached or already running on the new active node, it would be more economical to simply transfer the state. The different types of transporting the services have been extensively considered by both the active network as well as the mobile agent communities. The framework should support all migration approaches and allow the mobile service to select the preferred one.

The migration of the mobile service state (and potentially code) between different active node platforms and diverse network environments is an interesting as well as challenging problem. As new code loading mechanisms emerge, the complexity of the problem increases. Therefore, the framework needs to provide a mechanism/protocol that allows negotiation and deployment of an available code loading method among the active nodes (work in this area has been considered in [9, 19]).

A critical factor determining the viability of any mechanism aiming to support the migration of services from one network node to another is security. This accounts both for the protection of the confidentiality of the information carried as part of the service state, as well as the authenticity of the nodes and also that of the information (both

code and state) exchanged between the active nodes. It is unlikely that one active node will "accept" and execute code with predefined state unless it can trust the sending node and can check the authenticity of the transferred data. Therefore, the framework needs to provide appropriate mechanisms that allow active nodes to establish and verify trust relationships between mobile service components.

To avoid the limitations often seen in today's mobile agent systems, the framework must account for *micro-* as well as *macro-composition* of services. Very often the design of composite active services (such as the one described in section 1.1) follows the model of a loosely coupled distributed system. Other times a high level service is decomposed into code components interoperating through a node local composition model such as the one proposed as part of the LARA++ [4] active component framework. Both these cases illustrate what we refer to as macro-composition. The framework must be able to provide basic mechanisms or services through which the different service components can interact when required – typically in the control path. On the other hand, when the developer of a mobile service favours a modular design of tightly coupled components that implicitly trust each other, it should be possible to facilitate composition within an instance of the framework therefore favouring micro-composition.

Lastly, the framework requires appropriate authentication and access control mechanisms to regulate access to individual functions of the framework API. Depending on the author and user of the mobile service, access to individual API calls must be controlled (for example, only authorised services should be able to update the *alternative active nodes* table). Ideally, these security mechanisms will be closely integrated with the node local access scheme to NodeOS services.

In order to develop a functional and extensible framework the design of the APIs and the exported functionality needs to be considered carefully. The goal must be a level of granularity that permits the active service developer to use the framework services at their full potential, while at the same time hiding the internal implementation details of each service. Wherever possible, a set of primitives must be used in order to abstract the underlying structure of one mechanism over another. To exemplify this requirement, consider the need for providing different ways of transferring state (and code), over the network, between instances of the framework. In most cases, the mobile active service developer should be able to call *"send (state_object, destination, my_credentials)"* without caring if ftp, http, or some other proprietary data transfer mechanism is used. In this way, as new data transfer mechanisms emerge, they can be integrated in the framework without breaking existing applications. On the other hand, for reasons of speed, efficiency, etc., a specific data transfer method may be preferred. Therefore, the API should provide ways of querying at run-time the existence of a specific method and select its deployment.

3 Design and Implementation

Founded on the requirements analysis presented in section 3, we developed a framework for mobile active service components as part of the research carried out for the ProgNet [5] project in Lancaster University. The current prototype is implemented as

an extension to a .NET [6] based execution environment (EE) running on the LARA++ active node [4] while ports of Java [7] and OpenCOM [8] are also under consideration as LARA ++ EEs.

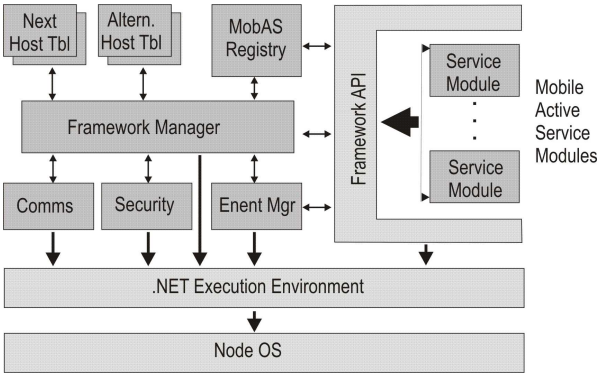

Figure 2. Framework Architecture

LARA++ [4] is component-based active router architecture based on the programmable switch paradigm. It uses a filter-based composition model particularly suited to supporting this framework and allows mobile components to be "hooked" directly into the forwarding path. The generic programming model provided by LARA++ enables our framework to support both in-band as well as out-band network programmability. This makes our framework suitable for control and management applications as well as services on the data path. Finally, LARA++ provides a filter-based pattern matching mechanism, enabling the flexible selection of packets subject to active processing. In this way it provides a generic solution of overcoming the efficiency and scalability problems of per-packet processing of active capsules.

3.1 Overview of Operation

A framework instance can host one or more closely coupled components that compose a *Mobile Active Service (MobAS)* at a micro-composition level. We herein refer to them as *MobAS modules* or simply *service modules*. If no MobAS modules are executing within the framework, then the framework operates in so-called *passive mode*, waiting for a MobAS from the network.

When an active application is implemented as a MobAS it initialises and executes within an instance of the framework. During the initialisation process (or at any time during normal execution) the MobAS may register with the framework a set of mobility events. These are typically application specific events or external triggers that depict certain system and/or network conditions.

Parallel procedures related to the MobAS (for example, population of the *next active node table*, callbacks for the registered events, etc.) can be either handled internally by the module that provides the actual service, or by separate module executing concurrently to the one providing the service – thus promoting a more scalable design

based on separation of roles. Furthermore, the various service modules can interact either directly or through the framework provided inter-module communication (IMC) mechanisms.

When a registered event, representing either an application specific condition, for example, a serviced user has moved, or an external condition such as resource exhaustion is triggered, its callback signals the *framework manager* to initiate the MobAS migration.

The implemented mobilisation mechanism uniformly supports intended and forced mobility. Since the service should be suspended while in consistent state, the framework always tries to allow the MobAS time to continue until it reaches a point where execution can be safely suspended. The MobAS developer can select and indicate such points in the program by using appropriate annotations. Then, control is passed back to the framework that takes action to package and send the MobAS.

Depending on the event source (application specific versus system/network related events) the framework must choose, either from the *alternative active nodes table* or the *next active nodes table,* a prospective host for the MobAS. Once a suitable active node has been selected, the framework installs, configures, and activates a new copy of itself on the chosen node using the ASDP [9] based service deployment interface.

Finally, when everything is ready the framework packages (serialises) the MobAS with its current state, adds a digital signature and if required encrypts it; the packages are then sent to the remote active node using a commonly available, and previously agreed, transmission mechanism. The framework instance at the remote end, will receive, the MobAS, (decrypt it, if required), verify its authenticity using the local node's public key, and resume execution where it was previously stopped.

Depending on the migration model the above process may vary slightly: i.e. only the service state may be serialised and sent to the remote active node where it is used with a freshly activated installation of the MobAS.

3.2 Architecture

As illustrated in figure 2, the framework consists of several building blocks each facilitating a distinct functionality. The proposed architecture augments the functionality provided by the EE, while regulating the access to the EE API. Figure 2 also shows the interactions among the distinct parts of the framework as well as between the framework and the EE/NodeOS.

- The *framework manager* is the "heart" of the framework. It is responsible for the coordination and the interfacing between the framework components, thus promoting a modular design whereby any component can be replaced (or upgraded) independently without impacting the integrity of the rest of the framework.
- The *MobAS registry* is the component responsible for the registration of the MobAS components with the framework. It holds the configuration of the MobAS regarding its operation (for example, service module scheduling priorities), and the MobAS user/author credentials that determine the API access privileges.
- The *event manager* is the component that handles the mobilisation events. Both application specific as well as external system and/or network events that can

trigger the migration of the MobAS are registered with this component along with any callbacks (that will be called upon firing an event).

- The *communication/control broker* component is responsible for the communication with instances of the framework installed on remote active nodes. Furthermore, this component handles the transmission of the MobAS from one active node to another, by negotiating a suitable data transfer method that is available on both active nodes. This is facilitated by means of a service deployment interface based on ASDP protocol [9].
- The *security broker* is responsible for providing the mechanisms to secure the code/state transmission between two active nodes. It holds a copy of the local node's private key and maintains a cache with the public keys of the active nodes that have been contacted in the past. It provides digital signing and encryption services for the framework.
- The *alternative hosts table* and *next hosts table* data structures maintain lists of potential future hosts for the MobAS. The *next host table* is maintained by the MobAS and holds a list of active nodes that may host the MobAS when an application specific event is triggered. The *alternative host table* on the other hand, is maintained by the framework (currently populated from the contents of a configuration file), and holds a list of alternative neighbour active nodes willing to host the MobAS.
- Finally the *framework API* controls access to the framework and the EE services.

4 A case scenario

To demonstrate the intended functionality of the framework, in this section we describe an example scenario whereby a server in the internet is assumed to be a victim of a DDoS SYN flooding attack [10]. It should be noted that with this example we do not aim to propose a comprehensive solution for the specific problem, but rather to exemplify the functionality of our framework.

Under this attack strategy, one or more malicious machines in the Internet send TCP SYN packets with various spoofed source IP addresses, at a very high rate to a victim server. The server replies with a TCP SYN/ACK packet to the spoofed IP addresses and waits for the ACK response to establish the TCP connection. Since the IP addresses are spoofed, and since the receivers of the SYN/ACK packets have not initiated the connections, the packets are dropped, leaving the victim node with a set of dangling TCP connection requests waiting to time out. The effect of this condition when it takes place at high frequency is the backlog queue exhaustion that leaves the server unusable as there are so many TCP connections waiting to be established.

Fighting against such types of DoS attacks is particularly tedious as it is impossible to filter the spoofed packets based on their source IP address. The most common countermeasure is to perform "traffic shaping" at the nearest to the victim router. Although this salvages the victim server resources, it usually cannot prevent the disruption of the service as new incoming connections are difficult to establish.

To counteract this attack we propose the use of a mobile active component that ultimately aims to track the source of the attack by following the spoofed traffic flow

from the victim to its source, based on the exhibited traffic pattern (of the attack) [12]. Then, "move" as close as possible towards the attacking host and install a firewall to block the malicious traffic.

The overall functionality would be provided by three cooperating mobile service components: (i) one that performs in-line network measurements to detect congestion patterns in the network (herein called measurements component), (ii) another one that samples the network traffic and performs SYN flooding detection (herein called flooding detection component), based on techniques proposed in the literature (such as SYN-FIN differentiation [13]), and (iii) one that processes the measurements of the other two components, classifies them, identifies the traffic pattern of the attack, and migrates to the next active node closer to the source of the malicious traffic (herein called DoS tracker component).

Initially, the DoS tracker component is installed in an active node close to the victim server. It then instruments the installation of several flooding detection components in neighbour active nodes and receives periodically the feedback from them. Once a DoS SYN flooding attack has been detected by the flooding detection component, the DoS tracker deploys within a range around the "abused" network interface, several measurement components and tries to identify the traffic pattern of the attack and find the originating point in the immediate network neighbourhood. If such a point is located then the DoS tracker "clones" itself on an active node closer to that network location. The same process is repeated again and again and the DoS tracker component "crawls" through the network closer to the source of the attack; for as long as there are available active nodes. When it cannot progress anymore or it has reached the network of the attacker, it can install a filter to block the malicious traffic as close as possible to the attacker's machine.

This scenario advocates the usefulness of the active service mobility to address problems that are otherwise difficult to tackle with conventional approaches, and demonstrates the functionality of our framework.

5 Related Work

Traditionally, service mobility has been facilitated by means of (mobile) agents. Their applications cover a wide range of distributed services ranging from processing of scientific data [13], evaluation of security algorithms [14], scalable network management [15], and so forth.

The development of distributed services is often supported by middleware solutions such as .NET [6], Java [7] and OpenCOM [8] that provide abstraction of system and network services (reflective middleware), portability, type safety, and other high level facilities that ease the development of distributed applications. As these platforms often don't provide complete support for developing mobile agent applications, a few frameworks have been proposed [16], [17] that extend or enrich the functionality of the aforementioned middleware platforms accordingly. The approach presented in [17] is perhaps the closest related to our work in this paper; yet like all mobile agent solutions, it accounts for user space applications, and cannot support the development of network services that can be deployed on the forwarding path.

On the other hand, the emergence of active networks research has enabled alternative ways of developing (low level – even within the forwarding path) mobile services. Certain classes of network applications such as traffic flow management in ad-hoc networks [18] can be efficiently developed by means of active capsule based solutions [1, 2, 3]. However, as we have argued in this paper capsule based solutions exhibit limitations with regard to their flexibility, applicability and the scalability outside the scope of their original application domain. On the other hand, programmable network approaches, like [4, 11, 20], have not considered the issue of generic run-time service mobility, apart from the simple case of remote loading/installing of code out-of-band.

6 Conclusions

In this paper we have presented the requirements analysis and the implementation of a framework for the development and management of mobile active services.

Initially, we outlined the need for service mobility in order to support mobile users and facilitate resilient and resource efficient servicing, in unreliable network environments. We then, examined the functional requirements of a generic framework that would enable the development of mobile services vertically across the network stack, in-band as well as out-of-band in the data path.

Finally, based on these requirements we presented an implementation that takes advantage of the flexibility and efficiency offered by the LARA++ programmable node platform, in combination with the portability and modularity provided by middleware platforms such as.NET, to deliver a generic, extensible and flexible development platform. The proposed framework overcomes the limitations encountered in mobile agent and active capsule technologies and provides a more viable, practical and complete solution (combining the best of the two worlds), which can be used to develop user space applications as well as low level network services targeting the data, control or management planes.

References

[1]. D.J. Wetherall, J. Guttag, and T.L.Tennenhouse, “ANTS: A toolkit for building and dynamically deploying network protocols”, Proceedings of IEEE Openarch, April 1998.

[2]. M.W. Hicks, P. Kaddar, J.T. Moore, C.A Gunter and S. Nettles, “PLAN: A Packet Language for Active Networks”, In Proceedings of 3rd ACM SIGPLAN International Conference on Functional Programming, pages 86-93, 1998.

[3]. B. Schwartz, et al. Smart packets: applying active networks to network management. ACM Transactions on Computer Systems, 2000.

[4]. S. Schmid, J. Finney, A. Scott and D. Shepherd, “Component-based Active Network Architecture”, in Proceedings of 6th IEEE Symposium on Computers and Communications (ISCC), Tunisia, July 2001.

[5]. “Programmable Network Support for Mobile Services”, Research Project funded by EPSRC Programmable Networks initiative, Lancaster University, 2001.

[6]. “Microsoft.NET Overview White paper” http://www.microsoft.com/technet/itsolutions/msit/dotnet.mspx, January 1, 2002

[7]. J. Gosling and H. McGilton. The Java language environment. White paper, May 1995. Sun Microsystems, 2550 Garcia Avenue, Mountain View, CA 94043, USA.

[8]. Coulson G., Blair G.S., Clarke M., Parlavantzas N., "The Design of a Highly Configurable and Reconfigurable Middleware Platform", ACM/ Springer Distributed Computing Journal, Vol. 15, April 2002.
[9]. M. Sifalakis, S. Schmid, T. Chart, D. Hutchison. "A Generic Active Service Deployment Protocol". In Proceddings of 2nd International Workshop on Active Network Technologies and Applications (IECE ANTA), p. 100-111, Osaka, Japan, May 2003.
[10]. CERT advisory ca-1996-21. "TCP SYN flooding and IP spoofing attacks". http://www.cert.org/advisories/CA-1996-21.html, Sep 1996.
[11]. A. Galis et al. "A Flexible IP Active Networks Architecture". Proceedings of 2nd International Working Conference on Active Networks (IWAN), Tokyo, Japan, October2003.
[12]. H. Wang, D. Zhang, and K. G. Shin. "Detecting SYN flooding attacks" in Proceedings of IEEE Infocom'2002, June 2002.
[13]. E. Korpela et al. "SETI@home: Massively Distributed Computing for SETI". IEEE magazine "Computing in Science and Engineering". January 2001.
[14]. http://www.distributed.net/projects.php
[15]. A. Bieszczad, B. Pagurek, and T. White. "Mobile agents for network management". IEEE Communications Surveys, September 1998.
[16]. J. E. White. "Telescript technology: Mobile agents". General Magic white paper, 2465 Latham Street, Mountain View, CA 94040, 1996.
[17]. N. Suri et al. "An overview of the NOMADS mobile agent system". Proceedings of 6th ECOOP Workshop on Mobile Object Systems, Sophia Antipolis, France, June 2000.
[18]. S.-B. Lee, G.-S. Ahn, X. Zhang, and A.T. Campbell, "INSIGNIA: An IP-Based Quality of Service Framework for Mobile Ad Hoc Networks," J. Parallel and Distributed Computing, special issue on wireless and mobile computing and communications, vol. 60, no. 4, pp. 374-406, April 2000.
[19]. Prince D., Scott A., Shepherd W.D. "On Demand Network Level Service Deployment in Ad-Hoc Networks". Proceedings of 8th Conference on Personal Wireless Communications, Italy, September 2003
[20]. R. Keller et al. "PromethOS: A Dynamically Extensible Router Architecture Supporting Explicit Routing". Proceedings of the 4th Annual International Working Conference on Active Networks (IWAN), Zurich, Switzerland, December 2002.