Interference and Communications among Active Network Applications

Luca Delgrossi[1], Giuseppe Di Fatta[2,3], Domenico Ferrari[1], Giuseppe Lo Re[3]

[1] CRATOS, Universitá Cattolica del Sacro Cuore,
via Emilia Parmense 84, 29100 Piacenza, Italy
ldgrossi, dferrari@pc.unicatt.it
[2] ICSI, International Computer Science Institute,
1947 Center Street, Suite 600, Berkeley, CA 94704-1198, USA
difatta@icsi.berkeley.edu
[3] CERE, Centro di studio sulle Reti di Elaboratori, C.N.R.,
viale delle Scienze, 90128 Palermo, Italy
difatta, lore@cere.pa.cnr.it

Abstract. This paper focuses on active networks applications and in particular on the possible interactions among these applications. Active networking is a very promising research field which has been developed recently, and which poses several interesting challenges to network designers. A number of proposals for efficient active network architectures are already to be found in the literature. However, how two or more active network applications may interact has not being investigated so far. In this work, we consider a number of applications that have been designed to exploit the main features of active networks and we discuss what are the main benefits that these applications may derive from them. Then, we introduce some forms of interaction including interference and communications among applications, and identify the components of an active network architecture that are needed to support these forms of interaction. We conclude by presenting a brief example of an active network application exploiting the concept of interaction.

1 Introduction

The last few years have seen the growth of active networks as an innovative technology in computer networking. Traditional computer networks allow their users to share network bandwidth as a common resource. Active networks focus not only on bandwidth but also on other network resources, such as the computing and storage capabilities at the end-systems and intermediate nodes. It also provides the means to inject user code into these nodes, thus enabling user customisation of network protocols and services. The first consequence of active networks is that the entire network paradigm is destined to change. In this new scenario, traditional network protocols look static, rigid, and barely suitable. A new base protocol is required that allows for an efficient management of network resources and the delivery of user data and code. Traditional data packets are to be replaced by active packets (also called capsules), which can carry both user data and code. Intermediate nodes of an active network need to have the means to load and execute user code in an appropriate execution environment. This new network paradigm opens many interesting possibilities and at the same time it poses a number of new challenges. Modularity and extensibility represent the more general properties of active networks. Users can develop specific algorithms to be integrated in the network protocols, to achieve application-oriented network functions. Such functions can be injected into the network at run-time, based on specific needs. An extreme flexibility of the network can be achieved because single packets can specify their own management functions. Packets that do not contain code, as traditional ones, are forwarded by the default network functions. Although active networks do not increase the domain of deterministically solvable problems, they allow for technical solutions that are particularly suitable for some distributed applications. In several cases, this technology makes feasible network applications that would otherwise be not efficiently realisable. The extreme flexibility of active networks has caused the proliferation of many definitions for them, such as: programmable interface for the network [9], programmable network [15], adaptive protocols [13], platform for user-driven customisation of the infrastructure [12], network as a computational engine [7], and so on. Active networks require strict homogeneity of the basic software components in the network. Thus, an active network can be considered as a global distributed system. Active network technology introduces serious security problems. As user code can be executed on intermediate systems, active networks architecture should guarantee safety and security for user applications, nodes (both end and intermediate systems), and the network as a whole. Different architectures have been proposed in a recent past, which propose partial solutions to this problem [2], [5], [16]. The comprehension of the benefits that an active network implementation can produce is a subject that deserves studying. An efficiency measure has been proposed in [6]. The aim of this work is to focus on the interactions among active network applications. This seems to be an interesting potentiality of the active network paradigm, currently not explored in the literature. We feel that an efficient active network architecture should provide the means to allow applications to exploit the advantages that can be derived from the desired forms of interaction, and the mechanisms to protect applications against undesired forms of interaction. In the rest of this paper, we discuss a number of active networks applications and highlight the potential benefits they can gain from active networking (Section 2); we present some basic forms of interactions among applications including interference and communications (Section 3), and discuss some issues related to the introduction of mechanisms to support application interactions in an active network architecture (Section 4); finally, we conclude by presenting an example of an application exploiting the concept of interference (Section 5).

2 Active Networks Benefits

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,
- data processing capability along the path,
- adoption of distributed strategies,
- easy development of new network services.

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 customisation 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 optimise bandwidth utilisation, or, again, adapt the data format to different user specifications. Audio and video conferencing systems have been proposed in [4], [5], 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 [13], stock quotes and on line auction applications [15], distributed firewalls [13], and the distributed management of multicast trees [10].

A particular application proposed in [14] is an ad-hoc mobile firewall, whose aim is to inhibit the annoying denial of service attack known as the SYN-Flooding attack. The defender injects into the network a defence mobile agent that is able to recognise the intruder packets and to stop them in intermediate nodes closer and closer to the attacker's node. A defence based on active networking is inherently more powerful than a common attack strategy, and this case is interesting because it shows a typical interference that makes of an active network such a powerful tool that it becomes potentially dangerous. If no limitations were provided to the interference among applications, the resulting chaos would determine the biggest possible denial of service. From these considerations, we see that the design of an active network architecture capable of providing mechanisms that allow applications to declare their own desired degree of interference is essential.

Easy Development of New Network Services - 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 IETF *diffserv* working group is proposing an architecture [8] to support different services other than best-effort forwarding. Per-Hop Behaviours (PHB) are the bricks with which these new services can be built. Per-Hop Behaviours are dynamically allocated in the network nodes and the active networks 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 the path of unicast transmission is substituted by several parallel paths.

3 Active Networks Applications

As we pointed out in the previous sections, researchers have already been able to identify a number of applications that can potentially take advantage of the main active networking features to fulfil their tasks in an efficient manner. Here, we try to sketch a rough classification of these applications, based on a number of relevant criteria. The goal is to identify the essential components of a generic active networks architecture in which all type of applications fit well. The adopted criteria comprise the type of execution environment present in the intermediate nodes and the ability to share data among different applications.

3.1 Capsules

We start by describing what we feel is the simplest possible scenario. An application generates an active packet containing user data and executable code (capsule). The capsule is sent over a computer network and traverses a series of nodes - some of which are active nodes - on its way to the final destination. When the capsule reaches an active node, the software on the node identifies the presence of executable code, loads it into memory and executes it in a suitable environment. The results of this execution are then stored back into the capsule and the capsule is forwarded to the next hop towards its destination. This behaviour may be replicated at each active node until the destination has been reached. During the execution, it is possible to access critical information stored in an active node. For instance, a capsule may need the current value of a timestamp or detailed information on current traffic load, or on which routes are currently available for data delivery. In a reservation-based system, information on the amounts of resources still available and on the quality of service that can be obtained under the current traffic conditions could be made available as well. Thus, a scheme is needed that provides the means for a peer application to extract critical information from the network in an effective way. Even if this scenario looks simple, we can derive some conclusions from it. The problem of defining an efficient way for capsules to deliver results to a peer application has not been yet investigated. A first approach consists of letting the capsule reach the peer application at the destination: the application may then read the results of the execution directly from the capsule. As an alternative, a capsule may occasionally send back to the peer application at the sender side the results collected. If we push this idea a little forward, we can imagine applications injecting capsules into the network that never really reach a final destination but keep travelling and reporting results and collected data back to the sender application. This last scenario calls for the definition of a communication protocol between the application and its capsules. Although it would be possible for each application to define its own protocol, it would be convenient to define a common protocol to be used by all applications, so that nodes in the network can be used to improve the communications: for instance, an active node could resend a packet generated by a capsule if it can detect that the packet has been lost and the capsule has already left the node.

3.2 Interference among Applications

In a more sophisticated scenario, a capsule injects into an active node code able to modify the node's behaviour by the execution of one or more customised functions. These functions could be designed to manage (forward, discard, filtering, modify) the data packets that traverse the node. This can be accomplished either by modifying the behaviour of a task run by the active node (the assumption is that the active node provides the means to do this) or by creating a new task running in the node. The latter case corresponds to the activation of a new agent, e.g., a packet filter, on the node. This mechanism can be used by an application to modify the router's behaviour with respect to the data packets that will be next sent by the application itself. For example, it may be used to discard packets logically belonging to a substream when delivering hierarchically encoded digital video. Therefore, an application has the means to differentiate the service it receives from the network on a per-packet basis and dynamically adjust it. Although most of the proposed applications limit the agent actions to packets belonging to the same application which installed the agent, in some cases the action might be executed on packets generated by other applications, that may be unaware of it. We call this inter-application interference. Interference among applications can be a very powerful way of exploiting the active networking paradigm. However, it is necessary to provide a framework with strict rules that regulate interference and prevent illegal use by unauthorised applications. Also, appropriate mechanisms have to be built into the network to enforce these rules. We feel that an active networks architecture should provide means to:

- uniquely identify applications and capsules within the network,
- associate a set of routines and a memory area in the active node with the incoming capsule,
- allow for secure management of application interactions.

Some mechanisms have already been proposed in the literature. ANTS proposes the introduction of fingerprints to authenticate the application and the packet [16]. Such a mechanism is devoted to guarantee that capsules are associated with the correct environment (functions and data) in the Java Virtual Machine. This authentication procedure is aimed at the binding of the programming environment, and indirectly at security. Switchware provides a more general and complex authentication scheme, where the main goal is security [2]. Even in this case, no attention is paid to the regulation of interactions. In the example of the defence against the SYN attack we can imagine that the defender agents present some credentials to the intermediate nodes, whereas the attacker packets contain spoofed addresses. Although applications should be protected against an undesired interference, safe and reliable interactions remain a powerful tool to build effective applications. A number of "good" interactions could be set-up between ISPs, which could collaborate by sharing their services. Forms of interaction can be identified in the web proxies. In the stock quotes and on line auction applications, the fundamental requirement is that the user (client application) be able to trust the intermediate agent (injected by the server application), i.e., a mechanism of authentication is required. Any external action on an application data flow should have been previously accepted or, even better, declared by the application itself. Only such severe rules can allow the correct use of interference as a useful network service. We propose three possible levels of interference. They correspond to the degree of intervention that an application is willing to accept:

- no interference: an application may require a high security level for its data, and, consequently, it does not accept any interference on its packets;

– intra-application interference: applications using interference as a tool to achieve their specific goals could require an authentication mechanism that guarantees against intrusions from and by other applications;
– inter-application interference: this is the general case, where an application could accept that other network entities access and modify its own packets. For instance, this is the case of some network services shared by more applications, such as the traditional routing functions (able to access but not to modify packets), policing functions (able to discard some packets), encryption functions (able to access and modify packets), and so on.

An efficient active networks architecture should support all three kinds of interference among applications.

3.3 Communication among Applications

Communication between two or more active networks applications is determined by a mutual will to exchange information. Two or more applications, running on an active network, can establish real communication sessions, or more simply they can exchange some messages between each other. An analogy with traditional communicating processes is possible. When two applications decide to exchange some information, they can send some messages to each other, by adopting a common format. Many authors describe active networks applications as network protocols [1]. The communication between applications, in turn, should be managed again by a protocol, which will appear as a communication protocol between some other protocols. Typically, each application provides for some entry points, through which it can receive some information, or information requests by other active applications. Differently from the interference case, in the communication scheme a deterministic behaviour is entirely preserved and guaranteed. In this case, the provision and the management of possible interactions is up to the application. Instances of applications emerging in communication activities could be general-purpose utilities providing services on the network, which can be used by means of well-known handles. As we did for interference, we propose three different levels of communication:

– no communication: an application executes its code without requirements of external data;
– intra-application communication: this is the case of a multitasking application, where different components of the same application can exchange data among them;
– inter-application communication: two separate applications agree upon the exchange of data between them; to this end, they can use messages or they can set some shared variable located on intermediate nodes of the active network.

While communication appears as a mutual interaction, on the other hand interference involves a passive behaviour on the part of one of the subjects.

4 Architectural Issues

4.1 Intermediate Nodes

Several active networks architectures are currently under development in industry and academia. Different directing principles are underlying these projects, and as a result some of them present diametrically opposed characteristics of design. In this section we will focus on the main characteristics of intermediate system execution environments. Using some concepts derived from operating system design principles, the architecture of an intermediate system, which provides processing capabilities for user code, should present a layered architecture to guarantee different flexibility, security, performance, and usability levels. Only few existing active network architectures have adopted such a criterion as their design principle. Most of them provide only an execution environment obtained on top of pre-existing architectures, and delegate the facing of security problems to the language adopted for active codes. Another of the main topics, which differentiate existing architectures, is the entity that should be considered as an atomic object. Packets or streams can be adopted as the individual repositories of the actions of active codes. This corresponds to two different interpretations of the active networking concept:

- the first interpretation reconsiders the network protocol concept, by extending the control information contained in the packets with small pieces of codes. The packet code will be executed at each node and will process the packet data along the path towards the destination;
- the second approach considers the intermediate systems in the same way as an end-system. Standard functions or codes previously installed at intermediate systems constitute pipelines for data flows.

To take advantage from all the capabilities, some architectures adopt both approaches. Switchware [2], for instance, provides active packets containing mobile programs, and, at the same time, active extensions, providing services on the network elements which can be dynamically loaded. The unresolved problems in the design of an active intermediate system are still many. Nevertheless, most of them are similar to the problems of designing a multitasking operating system. Active technology transforms the intermediate systems from special purpose devices to shared general purpose computing engines. This evolution coagulates specific problems of network and operating system fields in a more complex situation to be faced. Problems such as active program naming [3], active node resource management, protection of active applications, and system integrity are common subjects of both research fields. Furthermore, the introduction of interaction capabilities between active applications entails the classical problems of inter-process communication. A language capable of providing the communication primitives must be adopted, and the system has to provide the necessary abstractions to accomplish them. The programming language adopted for the active codes is an other fundamental aspect of an active network architecture.

Its characteristics are, in some aspects, complementary to the active node operating system characteristics, because it makes up for the lack of operating system with its capabilities. These can be summarised in a strong type control and in the capability of static program verification before the capsules are injected into the network.

4.2 Security

Security problems are closely related to interaction activities. Intrusion in a private data flow is an undesired form of interference. An active network architecture allowing for application interactions must provide strong protection forms to guarantee correct and secure executions. Current active networks architectures propose different and sometime complex solutions to the security problems. Some of them adopt traditional authentication methods, to securely identify packets or data flows, which are allowed to perform safe operations on the intermediate nodes. Among the existing architectures, which significantly take into account the security problems, the Secure Active Network Environment [3] presents the most effective solutions. More than the active network architecture, such an environment (or a secure active network) should provide severe rules for limiting the capabilities of an application for interaction with another. As a consequence, a safe control of interactions imposes some differentiation about who can act on whom. To this end, it is useful to operate a distinction about the two different interaction schemes proposed in this paper. The communication scheme redraws, in some aspects, a message-passing operating system architecture, which has been proposed as a useful model for network and distributed operating systems, one that is able to encourage distribution and, to some extent, security [11]. The different interference levels require a policy, which allows a stream to disclose itself to another one. A potential solution is supplied by the definition of different levels of protection for an active application. An active application may allow reading, writing and executing its own capsules, to nobody (no-interference), only to itself (intra-application interference), to some authenticated applications, or it can disable any protection form (inter-application interference). Authentication protocols by means of public and private keys, and digital signature algorithms can be used to guarantee security and coherence of the communications. The impartiality of intermediate node operating systems guarantees against unrecoverable actions such as the complete malicious discarding of packets. A rigid per-packet or per-flow authentication presents arduous scalability problems. The following different aspects of the security problems should, as a consequence, be taken into account.

Active Node Security - This aspect regards the protection of an intermediate node security from external dangerous actions. The means available to face these concerns are:

1. the adoption of a programming language with reduced capabilities: a user code cannot access directly node resources;

2. the layered organisation of the intermediate node operating system, which is a precondition for its integrity, or an isolated user code execution environment like the sandbox of JVM in [16].

Active Network Security - In this case, security problems are related to the network as a whole. If an active code generated and forwarded more copies of a packet without any limitation, the network would be flooded in a short time. Such a denial of service could be avoided by introducing some mechanism like the TTL in standard IP packets. When a packet is duplicated, its copies will share the original amount of TTL. Such a mechanism on one side protects the network from flooding, on the other side it constrains the active networks application capabilities.

Application Security - Active network applications must be guaranteed against undesired interference performed unintentionally or maliciously by other applications. To this end, active networks architectures should adopt more restrictive security mechanisms such as:

a) naming strategy (for the routines installed in an active node)
b) authentication and authorisation on a stream and packet basis.

5 The "Counter" Application

In this section, we briefly describe an application designed to solve the problem of counting the number of intermediate nodes present in a large network. If we consider the Internet, the problem of determining how many routers are actually present does not lend itself to a simple answer: Internet addresses cannot help solve this problem, because nothing in the address structure allows for an intermediate router to be distinguished from an end-system. This information could probably be obtained by means of a hypothetical agent located on a node of the network which recursively interrogates all the adjacency detecting Internet routers to be marked, explored, and counted. The active network philosophy may as well offer advantageous tools with which to address this problem. Nomadic agents exploring the network could partition and explore the network more efficiently than a fixed, static agent. The idea is that explorer agents are capable of duplicating themselves whenever a switching point is encountered. The creation of such an application raises a problem due to the growth of the number of capsules. Safety and security aspects of the whole network require that all the capsules injected into the network have a limited TTL, in order to avoid uncontrolled flooding. The solution provided here adopts three different types of active agents: Source, Base, and Scouts.

- the Source agent generates the whole application. It is in charge of creating the initial Base agents, collecting the intermediate results, and co-ordinating the node marking and cleaning process;

– the Base agents are responsible for the local actions carried out by the Scout agents. They act as local collectors, and have some control over Scout actions;
– finally, the Scout agents, which are light voyager capsules, discover, count and mark the routers in the neighbourhood of the generating Base.

Once a Base agent has been injected into an active node, it sends a Scout agent to each adjacent node whose distance is bigger than the current one, distributing its amount of TTL among them. It assigns to the Scouts a maximum exploration distance. It then waits to collect the partial counters obtained by the Scout agents. When a Base agent obtains the partial results from all of its Scouts, it sends its collected value to the Source. The application should keep track of the nodes that have been visited and those that have not been. To this end, the visited nodes are marked, thereby necessitating a successive cleaning phase. Each Scout agent injected into a node takes the following actions: if the node has not been already visited, it marks the node as counted, decreases its exploration distance value and generates as many capsules as there are adjacent nodes at a greater distance from the Source. When the Scout has reached a limit node, it will then install a new Base agent for the following expansion step, sending the address of this border node to the Source. If the node has already been visited, it returns to the parent node, it increases its parent node counter by its own counter, and decreases the number of open paths. The TTL value determines the dimension of the explored area. This area, which is explored by the Scout agents generated by a single Base, is called a Zone. The above application, of which we are building a first implementation, uses intra-application communication forms. Scout agents send the collected results to the Base agents, which in turn communicate with the Source. The same application can be partitioned in a spatial way. Different Autonomous Systems can agree on the possibility of separately enumerating their own intermediate systems. Multiple instances of the same application can exchange their final results to collect the global result. This last case represents an inter-application communication example.

6 Conclusions

Active networks move the control of some network functions to applications (end-to-end argument) and at the same time allow for the execution of some application components in the network. Factors such as performance downgrading and security may represent potential problems. The success of such a paradigm may depend on the fast diffusion of new network services and protocols, and on the development of new applications. Some characteristics of active networks have been analysed, which can produce a more efficient implementation of traditional applications. We feel that an important way to exploit active network features will be that of writing applications able to interact with each other. We discussed two forms of interaction: interference and communication. The former is a powerful tool, which requires ad-hoc security mechanisms. The latter does not impose strong security controls, albeit no applications, which employ such a concept, have been yet proposed. Finally, the active networks paradigm drives the static concept of network protocol towards a network operating system, capable of guaranteeing basic efficient connectivity and adequate security levels, and which makes the different network resources available to the applications.

References

1. Alexander, D.S., Arbaugh, W.A., Keromytis, A.D., Smith, J.M.: Safety and Security of Programmable Network Infrastructures. IEEE Communications Magazine, vol.36, n.10, October 1998, 84 - 92
2. Alexander, D.S., Arbaugh, W.A., Hicks, M.W., Kakkar, P., Keromytis A.D., Moore, J.T., Gunter, C.A., Nettles, S.M., Smith,J.M.: The SwitchWare Active Network Architecture. IEEE Network Special Issue on Active and Controllable Networks, vol. 12, n. 3, May-June 1998, 29 - 36
3. Alexander, D.S., Arbaugh, W.A, Keromyts, A.D., Smith, J.M.: A Secure Active Network Architecture: Realization in SwitchWare. IEEE Network Special Issue on Active and Controllable Networks, vol. 12, n. 3, May-June 1998, 37 - 45
4. Baldi, M., Picco, G., Risso, F.: Designing a Videoconference System for Active Networks. Proceedings of the 2nd International Workshop on Mobile Agents, Stuttgart, September 1998
5. 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
6. 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
7. Bhattacharjee, S., Calvert, K.L., Zegura, E.W.: On Active Networking and Congestion. Technical Report GIT-CC-96-02, College of Computing, Georgia Tech.
8. Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W.: An Architecture for Differentiated Services. Internet RFC 2475, December 1998
9. Calvert, K.L., Bhattacharjee, S., Zegura, E.W., Sterbenz, J.: Directions in Active Networks. IEEE Communications Magazine, vol.36, n.10, October 1998, 72 - 78
10. Li-wei, H.L., Garland, S.J., Tennenhouse, D.L.: Active Reliable Multicast. IEEE INFOCOM'98 San Francisco, USA 1998
11. Nutt, G.J.: Centralized and Distributed Operating Systems. Prentice Hall International, 1992
12. 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, 80-86
13. Tennenhouse, D. L., Wetherall, D.J.: Towards an Active Network Architecture. Computer Communication Review, Vol. 26, No. 2, April 1996
14. Van, V.: A Defense Against Address Spoofing Using Active Networks. MIT Master's thesis, May 1997
15. 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
16. 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

This article was processed using the LaTeX macro package with LLNCS style