Proceedings of the 2nd International Conference on Advanced Communication Technology - 2000

Agent-populated Active Networks

Stamatis Karnouskos

German National Research Center for Information Technology,

Research Institute for Open Communication Systems (GMD-FOKUS)

Kaiserin-Augusta-Allee 31, D-10589 Berlin, Germany.

karnouskos@fokus.gmd.de

Abstract **¾ Active networking aims at transforming passive data carriers to active, dynamically configurable machines that no only pass data to each other but also perform computation on those data. In such a framework mobile agents can help either as carriers of the active code to be installed on active nodes or even as actors by managing the node autonomously. This paper explores the applicability of the agent technology in the active network (AN) domain. We present an architecture of an active node with embedded agent technology. Then we set the requirements for an AN language and finally we try to discover the benefits that this approach offers. At the end we give a scenario that illustrates a practical working notion of such an architecture.**

Keywords **¾ Active Networks, Mobile Agents, Security, denial of service, AN requirements and benefits.**

1. Introduction

The basic idea of AN [7] is the movement of service code which traditionally has been placed outside the transport network, directly to the network's nodes. Intermediate nodes should be able to compute on data they receive based on their state and goals as well as on the external application's need. All these should be done in a highly dynamic manner so that service introduction and performance are optimized. Opening up the network nodes and states may sound exciting but is not a trivial matter. This paper concentrates on the *discrete approach* of active networking where service deployment is performed separately from service processing. In this context we propose the application of agent technology in ANs.

Software agents [11] are a rapidly multi-directional developing area of research since the early 90s. Yet research community has not been able to find a clear answer to the most popular question "What exactly is an agent?" and the debate still goes on. A general answer could be: Agents are software components that act alone or in communities on behalf of an entity and are delegated to perform tasks under some constraints or action plans. Mobile Agents (MAs) shatter the notion of client/server model and eliminate its limitations. They provide robust networks as the hold time for connections is reduced only to the time required to move an agent, the agent carries credentials and therefore the connection is not tied to constant user authentication, load balancing can be achieved as there is no request flow across the connection in order to "guide" the agent and respond to results, furthermore there are already standardization efforts defining interoperable interaction between agent systems [1]. Further standardization efforts and guidelines that boost the usage of agent technology exist also in organizations such as the Object Management Group [2] and the Foundation for Intelligent Physical Agents [3]. Agents are computer and transport independent (they depend only on the execution environment) and therefore promote interoperability among systems and software.

2. Agent-powered Active Node

In Figure 1 the active node architecture is pictured. The node architecture constitutes of :

A programmable router. The router is accessed via an API for dynamic programming of it resources. The open node interface represents the abstraction of the router resources ranging from computational resources (CPU, memory etc) to packet forwarding resources (bandwidth, buffer, etc).

The NodeOS. This is the operating system running on each node (router) in an active network. The NodeOS provides the basic functionality from which the EEs built the abstractions presented to the active applications. The architecture of the NodeOS and its functionality is outlined in detail by the AN Node OS Working Group [5].

Execution Environments which are on top of the NodeOS, making use of its services. As noted [6] the functionality of the active network node is divided among the Node Operating System (Node OS), the Execution Environments (EEs) and the active applications. The architecture allows multiple EEs of various providers to coexist and be present on a single active node. Each EE (e.g. ANTS, ALIEN, Agent EE) exports a programming interface or virtual machine that can be programmed or controlled by 3rd party code. The NodeOS manages the resources of the node. One of the EEs is the Mobile Agent EE where agents execute when they visit the node. This EE is further analyzed in section 3.

Agents that reside in the Agent-specific EEs and via the facilities offered to them program the node. These can be either mobile agents (e.g. visiting agent) or even stationary intelligent ones that reside permanently in the EE and offer services e.g. tuning node's resources according to traffic, apply custom security schemes etc

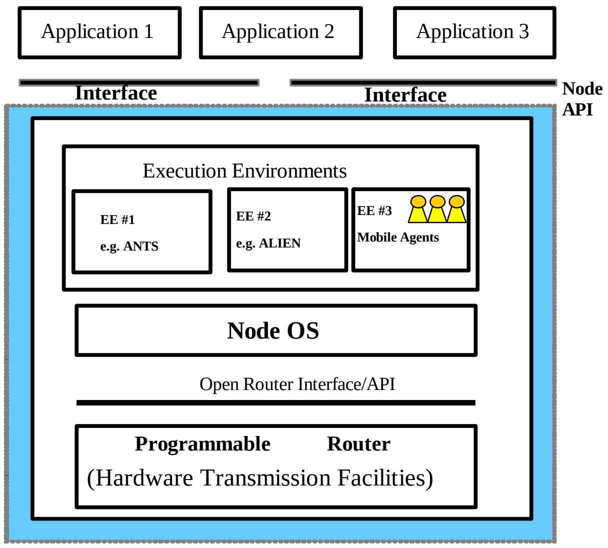

Figure 1. Active Node Architecture

The applications are able to access all the services offered by the EEs. Usually an application is bounded to one EE but we can foresee application that will take advantage of the various characteristics of EEs and possibly combine their services.

3. Agent EE Architecture

The agent system (Figure 2) consists of places. A place is a context within an agent system in which an agent

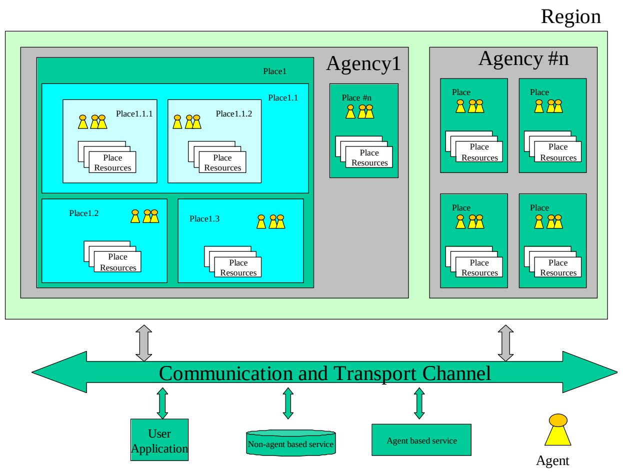

Figure 2. Agent EE Architecure

is executed. This context can provide services/functions such as access to local resources etc . A place is associated with a location which consists of a place name and the address of the agent system within which the place resides. Places can contain other places (nesting places). All places follow the parent-child paradigm of Unix processes in the way that each child is assigned/makes use of its parents resources. Also its policy is an extension/customization of its parent's policy etc. A Place can be used in different ways. Places are:

Dynamically assigned to agents as they enter the node based on some criteria e.g. all agents coming from a specific user or location or agents belonging to a specific policy scheme etc.

Statically assigned per entity (e.g. user, enterprise etc). Here static resources are given to the Place (after agreement with the node provider) and the local resource manager manages them. In this way it is possible for an enterprise to setup a network of places in various nodes, creating a Place-Oriented Virtual Private Network [15]. This offers several advantages e.g. secure communication between company-trusted places etc.

A Policy manager and a resource manager are assigned to each place and are given the general security guidelines, which can never be bypassed. If an agent has sufficient credentials then he can fully interact with the components e.g. change the place's policy, ask for more resources, insert elements into the component database etc.

The existence of different Execution Environments (EEs) for agents (which are called Places within the Agent architecture) that have the same owner/characteristics serves the need to avoid unwanted interactions. Isolation done by EEs is similar to the sandbox idea that exists in Java. Since in each place agents with common characteristics (e.g. of the same owner) are gathered the possibility of attacking each other is lower as usual. Of course advanced security facilities offered by the place can be used to minimize these risks. Furthermore if one wants he can use a place as a TestPlace and allow suspicious agents to execute there, monitor the results and then determine if he will allow them to execute in the real place. Certainly if you see for instance that an agent changes inappropriately the policy file of the TestPlace you forbid execution to the desired place (which otherwise would be catastrophic). Also agents are somehow isolated since each one has its own classloader.

Places beyond having unique IDs, also hold their own public/private keys. An agent can ask to be signed in order to have a proof that it passed via this place. This also helps with the so-called "multi-hop" security problem. If every Place signs a specific part of the agent then we can trace back partly or fully the route the agent followed. Based on that info we can take further security decisions. Let us mention that if there is one malicious host who tries to break the chain of signatures (not sign the part of the agent because he performed something maliciously and doesn't want to leave any traces) it will be detected by the next non-malicious place.

4. The Language Choice

Selecting a language for ANs is not a trivial issue. Tradeoffs between security and performance are critical parameters in the choice of a language especially if this language will be used by user-injected general purpose code. If ANs were to operate in a completely trusted environment then any modern programming rich in features language would be appropriate. But here we have to deal with a heterogeneous untrusted environment where the author of the injected code, the user of the code, the owner of the hardware, the owner of the execution platform can be different entities governed by different security policies and possibly competing interests.

AN's biggest problem is security. Thus all decisions in designing/operating it should be with security in mind. So we require a language that can have some special characteristics such as :

- ***Strong typing***. This means that a program cannot arbitrarily access the host computer's memory. Memory access is limited to specific controlled areas having particular representations. Thus in such a language common programming errors are avoided.
- ***Garbage collection***. Of course each user/agent can manage the memory (allocate/de-allocate) he is assigned. Lets suppose that he frees some memory blocks (memory looses his type) and that these blocks are reallocated to another agent. Then this agent is able to read the data on that memory blocks and acquaint info (possibly security critical) about the operations of the previous user of that memory space. With automated garbage collection we make sure to avoid such problems associated with dangling pointers.
- ***Access controllable module view*** allows us to view a module via multiple interfaces. Thus we may have a reach feature module that provides different capabilities to different users. In this way we are able to modify flexibly who can see/do what and how.
- ***Dynamic Loading*** is unquestionably a must! We want to make modifications and load new functions/capabilities while our system is up and running. It would be out of question to shutdown an Active Network router every time we want to update a software component or provide a new capability. Furthermore by dynamic linking it is easier to keep our system up-to-date since the latest version of code and libraries will always be used.
- ***Communication support***. The language should have its own optimized libraries for basic communication between the applications and of course network support. Object communication, programming with sockets, establishment of URL connections etc are mandatory in a networking environment.
- ***Widely used / evolvable*** is a non-technical characteristic of the language we need. This is not for commercial/political reasons but for practical ones. A language used by a small group of people might be task-specific but it would be difficult to advance and keep up to date. Also bugs, errors misbehavior would be seldom if at all reported. Thus we need a language that is widely used so that it evolves fast and day by day new features are added constantly depending on the needs.
- ***Platform independence*** is not mandatory but would be of great help since our efforts could be ported/deployed easily to a heterogeneous environment such as that of AN.

Having in mind all the above one could design a new language tailored to the needs of active networking and our system. A small sample of difficulties he/she would face is

- designing from the scratch a new language with a bunch of desired features as mentioned above (e.g. safety, performance),
- if we don't manage to address all required features needed by the user it would be impossible for user to implement the mobile code he wants,
- it would require a huge amount of work to keep the language up-to-date with increasing demands,
- it would be used by a limited number of people (AN people only) and therefore bugs, errors, etc would be seldom if not at all reported.

The other approach is to use an existing language. Java is not an AN language itself but covers reasonably our requirements. Java is a very popular language designed especially for mobile code and most important with security in mind. It supports dynamic code loading, concurrency, communication between networking applications (http, sockets, RMI etc) and basic security services (presence of a security manager). Java nowadays is used extensively not only in research domain but also in industry. Therefore bugs, errors are found and reported fast. As it is a commercial product it advances and day by day new features/libraries are added. Furthermore Java offers platform independence which is an significant factor as it assures portability within a heterogeneous environment such as the AN environment. That in addition with the support of object oriented concepts like polymorphism and inheritance make the development of active components easier, as these components are seen as an abstract object of code to be transported and installed in an environment no matter of the underlying architecture.

5. Expected Benefits

We think that AN community should adopt the agent approach as it offers many solutions and enhancements in key-feature areas that AN community looks for. Our approach is agent oriented not just for the shake of technology but because of the great benefits we can import from the agents to the AN technology. Some of them are:

- **Decentralization and Autonomy**

Many tasks/applications require a continuously open connection and a fixed network topology. Agents don't have that requirement and therefore ANs can benefit from it. Agents are able of working autonomously and in a decentralized manner. They exploit the locality and achieve optimization in the usage of resources they are offered in that location. Thus problems such as unpredicted network latencies in critical real-time systems (e.g. robots in a manufacturing process) can be avoided. Also by using agents we don't have to develop new transport mechanisms for the deployment of active components to the nodes.

- **Flexibility**

Users are able to launch agents and customize services easily. Most important the user doesn't have to be online all the time since he can send his agent and then disconnect from the network. The agent carries certificates from the originator and acts autonomously according to its internal goals. Furthermore agents provide mechanisms for monitoring, logging, updating etc which can ease tasks like administration/management of an AN node.

- **Adaptivity**

Changes in the environment in which an agent operates trigger possible changes to agent's behavior. Agent senses the environment, analyzes the new data and acts accordingly. E.g. a group of agents monitors the consumed resources in a network. These agents can exchange information and travel in the network in order to get a global view of the network's state. Depending on their goals as well as their capabilities they can interfere and e.g. change the routing tables of hosts in order to provide better exploitation of the bandwidth. This adaptivity promotes the optimal network service configuration and function.

- **Interoperability**

Network infrastructure is heterogeneous both in hardware and in software matters. Agents are computer and transport independent (depend only on the execution environment) and therefore promote interoperability among systems and software. It is possible with agents to implement interactions with any legacy systems and currently existing services and make it available to other heterogeneous agents e.g. via MASIF interface. Although the ground is pretty new, standardization efforts exist within many organizations e.g. Object Management Group, Foundation for Intelligent Physical Agents and will be available in the near future.

- **Security**

ANs allow users to inject their code to the node. That is a security critical activity. These issues have been addressed and solution exist within the agent community. Agent technology is capable of providing authentication, authorization, integrity-check and privacy mechanisms to AN managers so that they can have control over the network and its resources. Security is generally provided by exploiting application level services such as Public Key Infrastructure [9] and solutions/tools available today for applications. Agents act on behalf of a user/entity/enterprise etc and carry some credentials (e.g. signed by the owner or the originator etc) and based on these credentials and the local policy a security manager [4] could deny or allow an agent to access his system. Also as the research area of security is agent systems is a hot domain, we expect in the near future more and better security solutions. Those solutions can be automatically applied in the ANs.

- **Scalability**

This AN-based architecture is a decentralized one and can scale easily. AN nodes can host an agent system that could be low or high populated by agents that act and interact with each-other providing services and advanced features [10].

- **Safety**

The usage of Java as an implementation language offers some security and safeness level. Furthermore the idea of Places acting as sandboxes avoids unwanted interactions within an agent system.

- **Performance**

Our approach is based on Java. Java based systems for the moment lack performance. We expect this to change in the near future as efforts are being made in this direction. Nevertheless performance can be achieved/enhanced with other techniques such as component (agent code, protocol, algorithm etc) caching etc. From AN point of view intelligence is added at a network level offering the ability for exciting network wide applications that can configure each end every node for optimal application and overall performance. It has been recognized that network performance is not necessarily correlated with application performance [8]. ANs may perform actions that on first glance appear to degrade network performance (e.g. lower packet throughput) but actually they bring improvement to the application and to the network itself by reducing demand of bandwidth at end-points, reducing network congestion etc.

- **Robustness & Fault Tolerance**

Mobile agents are programmed with their internal goals and logic. Taking also into account their ability to react to the changing environment and unpredicted situations, it makes easier to design and implement robust and fault tolerant systems.

- **Software Independence & Evolvement**

Current distributed systems exchange data via a standardized way (protocols). Each node owns the hardware specific code that implements the protocol needed to communicate with the outside world. However the protocols and their supported features evolve and often we face the problem of outdated protocol versions which are inefficient and insecure. Agents on the other hand can help effectively in this problem. They can move to remote hosts and establish "channels" based on protocols that are task specific and not even standardized. Furthermore they can update node's components automatically, therefore keeping our infrastructure always up-to-date since all network components will be updated in parallel shortly after the announcement of a component update by the manufacturer.

The agent based approach allows for a high level system design including business aspects. The market is moving towards a service orientation and agents can fit well as service oriented software. Agents have a natural place in the application model as a) wrappers of legacy systems or as embedded smart systems, b) as powerful middleware that glues together distributed components, c) as intelligent and adaptable interfaces that support online/offline user interaction. Active networks is a good area to apply the agent technology as they will benefit on all the above mentioned sections.

6. Denial of Service Example Scenario

A hot issue in the AN research is the security one. On first thought, as AN open up the network, they seem to be more vulnerable to security threats than traditional networks. However the flexibility they add to the network itself can be used to deal effectively with traditional network attacks. We give here a scenario taken from the security domain that demonstrates how the agent based AN approach can lead us to a more secure network.

We will demonstrate how agent-based ANs can change dynamically the security of a node and promote network security (and not only on local nodes). Denial of service attacks are very common in current networks and are very difficult to be dealt with. The aim is to exhaust maliciously all resources of a node and therefore denying services to legitimate users. Such attacks can be initiated from a single host or multiple ones that reside within the same corporate network (a compromised trusted host) or are external .

Lets try to take one sort of denial of service attack and try to see how agent-based ANs can deal with it. We assume that node A (Attacker) is trying to attack node V (Victim) by opening aggressively thousands of connections to that node or by trying to flood all connections to that node with messages. The V node detects (via its policy or its intelligent stationary agent - lets call it Security Guard (SG)) unusual high number of network utilization and triggers immediately the attack-protection tactic. Immediately the SG changes the local policy database and further requests/flow from that node is denied access/ignored. Subsequently it dispatches child-SGs (agents with specific mission) which it multicasts to his neighbors (the attacker may have -and in this paradigm has- alternative routes to attack a node). Child-SGs have as goal to inform SGs on neighbor nodes and block/filter all traffic from A-node to V-node. Child-SGs are installed if permitted (or pass the necessary info to those node's SG) to those nodes and block again requests/flow coming from A-node and directed to V-node. Normal nodes that are not active just transfer the agents to the next node. Finally agents via this multicast reach the nodes that are directly connected with the A-node and block any further requests of this node.

Now this A-node can be an external or an internal node of a corporate network. Also maybe the node is a malicious node (the node's manager is not trusted) or maybe an element of the node is behaving maliciously or malfunctioning. If the node is malicious then there is nothing more that can be done. All requests/flow is filtered and blocked in the nearby nodes, thus protecting the rest of our network. If the node is not a malicious one but a component misbehaves e.g. an agent is behaving maliciously then it might be possible to install the blocking agent on the A-node and provide info to the A-node SG. Then the A-node SG (assumed not to be associated with the attacker) might identify the entity responsible for the attack (e.g. a malicious/malfunctioning agent) and take the appropriate measures e.g. kill it. If a component is behaving maliciously by error (e.g. a crash has corrupted the disk space where the component is stored and therefore the component itself), then that component could be replaced with a new version by an agent and the node would stop malfunctioning.

Action plans to be taken once the attack has been identified is a matter of policy, intelligence and capability of an agent. In any case the agent-based ANs have the

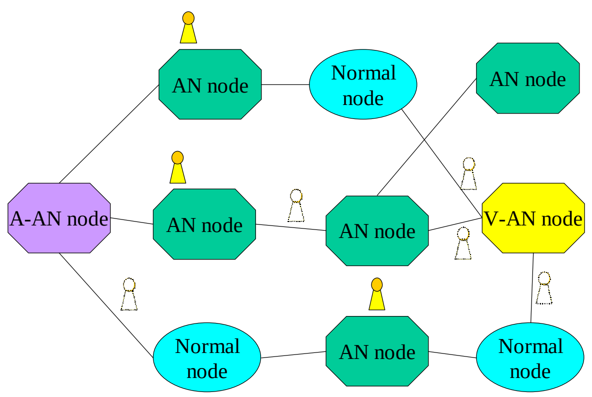

Figure 3. Denial of Service attack & blocking

ability to change dynamically the policy on the active nodes providing a more secure infrastructure and protecting not only the local network but the whole intranet. We note that normal nodes (such as that residing next to A-node in Figure 3) can't be protected/notified for this kind of attacks nor can they promote corporate network security as they act only for their own good and don't possess the means to interact with the other nodes.

Lately highly sophisticated tools for distributed denial of service attacks such as Stacheldraht [12], Tribe Flood Network [13] and Trinoo [14] have appeared. With such tools it is now relatively easy for a single person to perform a distributed attack from thousands of compromised hosts. We ought to say that fast and automated ways of identifying and dynamically blocking such types of attacks is not just desirable but a question of survivability for next generation networks. Internet nowadays is defenseless and a secure network that can quickly evolve its security provisions and protocols to keep ahead (or at least deal in an effective and quick way) of the attackers has to be the prime mission. We believe that the combination of agent technology embedded in active networks will be able to provide the flexibility and the services needed for a security aware network.

7. Summary and Conclusions

An agent-based active node architecture has been presented. This approach uses agents with different features e.g. mobile, stationary, intelligent, goal oriented etc to empower the current passive routers and to transform them to AN nodes. We have showed that Agent technology is a promising candidate for the development of Active Networks. It offers benefits to the users as well as the developers of ANs including flexibility, adaptivity, decentralization and autonomy, interoperability, scalability, security etc.

The outlined scenario shows a small sample of what kind of applications can benefit from the agent paradigm. Alternative VPNs [15] can be also deployed with minimal effort and low cost, suitable for small working groups that can't afford a legacy VPN. Security notifications can be delivered to nodes and update its databases and security threats can be effectively blocked near the source, protecting thus the rest of the network. In the future we intent to validate the above concepts by building an AN node with embedded agent technology [16]. By using an legacy IP router and enhancing it with an agent add-on we will demonstrate how it is possible to satisfy the ever increasing demand for more sophisticated and highly configurable/programmable services.

It is very likely that agent technology will play an important role in the development and expansion of the Active Networks. The basic characteristics of mobile agents such as mobility and autonomy can push networks to become more "open", active and more powerful. Agent technology advances continuously. In any case agent domain has made significant contributions in the area of code mobility and security and it wouldn't be wise reinventing the wheel every time in every approach we take. By integrating agents we make sure that in the future we will have all the time the state-of-the-art agent technology in our network and that simply means that our infrastructure will keep on advancing as long as it is connected with this parallel developing domain.

REFERENCES

[1] MASIF - Mobile Agent System Interoperability Facility, URL : http://www.omg.org/docs/orbos/98-03-09.pdf

[2] OMG Web Site : http://www.omg.org/

[3] FIPA Web Site: http://www.fipa.org/

[4] S. Karnouskos, I. Busse, S. Covaci ," Agent Based Security for the Active Network Infrastructure", 1st International Working Conference on Active Networks, Berlin, Germany, June-July 1999.

[5] Node OS Interface Specification. AN Node OS Working Group, Larry Peterson, ed., January 24, 2000.

[6] Architectural Framework for Active Networks, Draft version 1.0, K.L. Calvert, ed., July 27, 1999.

[7] A. T. Campbell, H. G. De Meer, M. E. Kounavis, K. Miki, J. Vicente, and D. Villela, "A Survey of Programmable Networks", ACM SIGCOMM Computer Communications Review, Vol. 29, No 2, pg. 7-23, April 1999.

[8] D. Wetherall, U. Legedza and J. Guttag. "Introducing new Internet services : why and how", IEEE network magazine, July 1998.

[9] Simple Public Key Infrastructure. URL : http://www.ietf.org/html.charters/spki-charter.html

[10] "Mobile Agents - Enabling Technology for Active Intelligent Network Implementation", Markus Breugst and Thomas Magedanz. IKV++ GmbH, Technical University of Berlin. IEEE Network May/June 1998.

[11] Cetus Links on Mobile Agents : http://www.cetus-links.org/oo_mobile_agents.html

[12] "The *stacheldraht* distributed denial of service attack tool", David Dittrich, University of Washington, 31 Dec. 1999. http://staff.washington.edu/dittrich/misc/stacheldraht.analysis

[13] "The *Tribe Flood Network* distributed denial of service attack tool", David Dittrich, University of Washington, 21 Oct. 1999. http://staff.washington.edu/dittrich/misc/tfn.analysis

[14] "The DoS Project's *trinoo* distributed denial of service attack tool", David Dittrich, University of Washington, 21 Oct. 1999. http://staff.washington.edu/dittrich/misc/trinoo.analysis

[15] S. Karnouskos, I. Busse, S. Covaci, "Place Oriented Virtual Private Networks", in the Proceedings of the Thirty Third Hawaii International Conference on System Science (HICSS-33), January 4-7 2000, on the island of Maui, Hawaii, USA.

[16] BANG - The Broadband Active Network Generation project. http://www.fokus.gmd.de/research/cc/glone/projects/bang/