# An Architecture for Publishing and Distributing Service Components in Active Networks

N. DRAGIOS, C. HARBILAS, K. P. TSOUKATOS, G. KARETSOS
School of Electrical and Computer Engineering
National Technical University of Athens
GREECE

*Abstract* - Application level active networks provide a way of transforming the current network infrastructure into one where new services and protocols are more easily adopted without the need for standardization. In this paper we deal with an application layer active networking system, and address the problem of publishing and distributing software components, provided by trusted service providers, throughout the active network. We propose a distribution architecture that is based on forming a network of dedicated servers providing Content Distribution Network (CDN) functionality; this leads to smaller response times to client requests and decreased network traffic. Experimental results from a test network configuration illustrate the benefits obtained from exploiting CDN capabilities, and support the viability of our approach.

*Keywords* – Active and programmable networks, content distribution.

## 1 Introduction

The evolving technology of Active Networking (AN) aims at changing the way computer networks behave. Activeness in networks is an idea proposed in [13], and seeks to address the problem of slow adoption of new technologies and standards, as well as the slow evolution of network services. Activeness entails injecting programmability into the network infrastructure, so as to allow the rapid introduction of many new services inside the network.

In this paper we focus on an application level active network. Here, processing of various data flows within the network takes place at the application layer (not at the network layer). References [6, 7, 9] discuss an Application Level Active Networking system (ALAN), which has been implemented to provide users with a flexible framework for supporting active services within traditional network boundaries. Software components implementing these services, called proxylets, can be loaded onto service nodes, called active servers, where they are executed on demand. This approach is similar in spirit to the Active Services framework of [1]. In this context, the distribution of service components, provided by trusted third parties, throughout the active network, is of considerable interest, for it directly affects response times to client requests and the volume of network traffic. We herein attempt to address this issue. Consequently, our focus is not on the AN per se, but on the network architecture used for publishing and distributing the service components throughout the active network, so as to better support the desired active functionality. We note that the idea of building Content Distribution Networks (CDNs) for enabling speedy, uninterrupted, and reliable access to web content may well fit in an application layer active networking environment, where service components are requested from specific web servers. We thus exploit the CDN concept to present a distribution mechanism that implements part of a CDN's functionality. The proposed architecture has been tested, behaves as desired, and leads to smaller response times to client request and decreased network traffic.

The paper is organized as follows: In Section 2 we describe the structure of the particular application level active network under consideration. In Section 3 we present the architecture for distributing and publishing service components; this is based on forming a Proxylet Distribution Network of servers. In Section 4 we report experimental results from a test network configuration which illustrate the benefits obtained from applying CDN technology and support the viability of this approach.

## 2 An application level active network

The application level active networking system presented in this section was developed in the

context of the European Commission IST project ANDROID (Active Network Distributed Open Infrastructure Development) [5] and borrows from the ALAN system discussed in [6, 7, 9].

In ALAN, clients can place proxylets onto service nodes, and execute them on demand. These nodes provide an Execution Environment for Proxylets (EEP), allowing these proxylets to be run. Requests can be sent to the EEP to load a proxylet, referenced by a URL, and the node will load the proxylet subject to checking the permissions, validity and security. Proxylets can be downloaded from a number of different sources, known as Independent Software Vendors (ISVs), and run on an EEP. In our approach, proxylets exist as Jar files, containing Java classes placed on a WWW server and can be referenced via its URL. They are self-describing, so that they can be used effectively in a dynamic environment. This is achieved by specifying appropriate proxylet metadata, expressed in XML, which include the proxylet functional characteristics, their facilities for communicating with other components and the corresponding security policies.

# 3 Publishing and Distribution of Proxylets

In this section we discuss the content distribution mechanism used for publishing and distributing the service components, and present in detail its architecture.

## 3.1 Content Distribution Network (CDN)

A CDN is an independent network of dedicated servers that web publishers can use to distribute their contents throughout the Internet [4]. Basic mechanisms supported by a CDN are caching, with all the advantages it offers, transparent routing of request to a server that can satisfy the request, and securing of the contents from modification. A CDN infrastructure can be divided in three main components: (a) the redirection infrastructure, (b) the content delivery infrastructure and (c) the distribution infrastructure [3]. The first one consists of the mechanisms utilized by the CDN to redirect the client request towards a server that contains a copy of the requested object. The second one consists of a number of servers, which deliver the requested objects and behave as content providers. Finally, the third one includes mechanisms for moving contents from the origin server to the servers of the content delivery system.

Our effort is focused on building an independent network of servers that implements part of the functionality of a CDN. We call this network a Proxylet Distribution Network (PDN). The PDN distributes the metadata of the available services, along with the actual services, i.e., the software components implementing those services. Those services are offered by ISVs, through web servers providing the proxylets. The hosts comprising the network, which provides CDN-like functionality, are called Proxylet-Brokers (PBs). These hosts serve as brokers, for they mediate between the clients and the content providers to inform the former about what services are available and bring them close to the latter in a transparent way.

In our approach, one PB is located in each administrative domain, where one or more active servers may exist. PBs could also be placed anywhere in the network, since their services are quite independent from any other procedure. The gathering and distribution of information about services could be seen as an off-line process, similar to the creation of a VPN on our AN, so PBs do not depend on any other component of the AN. Their network is built independently and its purpose is to serve the clients of the AN. However, PBs need to be close to active servers, where the proxylets are loaded and executed, and close to clients, who need to have quick access to information about the services available on the network. Guided by these considerations, we decided to place one PB in each administrative domain, so that they are as close as possible to where their services are needed.

## 3.2 Building the Proxylet Distribution Network

The Proxylet Distribution Network (PDN) is built in a step-by-step manner. The only information a new PB needs in order to join the PDN is the IP address of another PB that is already a member of the PDN. To this end, each time a new PB is launched at least one IP address of some other PB is passed to the new PB by the PDN administrator. The PDN administrator is responsible for making the IPs of participating PBs available to any one interested. Thus, a new PB contacts a member of the PDN and follows a join process. This takes place as follows: (a) the member of the PDN

accepts the IP address of the new PB and sends him the list of all IP addresses of the PBs participating in the network, (b) the member of the PDN also sends all information it has, if any, about services provided by ISVs in this network; what is actually sent to the new PB is the proxylet metadata.

After this join process, all PBs in the PDN, including the new one, are aware of the services available. Then, the new PB contacts all other members of the PDN and identifies itself; i.e., he sends his IP address to all other PBs, so that they add the new PB in their list of members of the PDN. The end result is a completely interconnected network of PBs, the Proxylet Distribution Network.

Using the aforementioned procedure, PBs may join or leave the PDN as needed, no matter their number. Whenever a new PB comes in, it informs the rest of the PDN about his arrival, whereas if some failure occurs and a PB cannot be contacted, it is removed from the list of members of the PDN. Thus, any number of PBs can access the PDN at any time, making the network of brokers a dynamic set that shrinks and stretches in an autonomous manner.

### 3.3 Proxylet metadata publishing

The next step is to see how the PDN is populated with proxylet metadata provided by ISVs. During this phase a new ISV presents itself to the PDN by contacting the closest PB in order to publish the proxylets it provides. The new ISV sends all proxylet metadata to the PB it contacted as a number of XML files, each one representing one proxylet. Now that these metadata are present on at least one network node, the PDN should distribute them across all PBs. The PB reached by the ISV contacts every PB of the PDN separately, and in turn distributes the metadata received from the ISV. After that, all PBs have the same view of the available services (proxylets) and associated metadata describing them.

Given that one PB resides in every administrative domain, where clients have access to the network and request services from, the gathering of proxylet metadata information close to the client results in a quick browsing of the available services. If the PDN facility were not in place, a client would need to know and query all remote ISVs, in order to find out if they own the required service or not. The difference in response times between these two cases is made apparent in Section 4, and the faster response gained from the PDN is what renders it important and necessary.

Although the replication of all XML files at all nodes of the PDN seems to be inefficient and consuming both of bandwidth and disk space, this is not really the case. Firstly, most of these file transfers are performed off-line, without aggravating the traffic of the actual active network, and, secondly, the size of these files is quite small, allowing them to be stored at all nodes of the PDN.

### 3.4 Service component request

The client component decides what service components an application needs, and asks the PDN to find and bring them close to him. The client is presented a list of all available proxylets, and their main characteristics, so that he selects the most appropriate service for his application. To this end, the client contacts the closest PB – the one residing at the same domain - and receives a list of all proxylets provided by the PDN, along with their metadata. The client either asks to check all metadata coming with a service, or requests the downloading of the service in his administrative domain. In the former case the PB responds with a complete list of attributes-values pairs contained in the XML file. In the latter case the PB establishes a URL connection to the ISV providing the proxylet requested by the client, and fetches the Jar file, i.e., the actual code for the service. When the Jar file is fetched it is cached locally and made available to any application, run by a local user, to use it by loading it to an active server. The PB that has just fetched a service from an ISV informs the rest of the PDN about this action, so that all other PBs know that the fetched service is available not only at the appropriate ISV, but also elsewhere in the PDN.

Smaller response times, reduced network traffic and reliability are achieved by this approach. Any other client can directly fetch an already cached service without downloading it from the ISV. Moreover, suppose a client contacts a PB and requests a service that is not cached locally in the PB. If this service is cached elsewhere in the PDN, it can be fetched directly from the PDN instead of the ISV. This leads to considerable reduction of traffic at the ISV web server, as well as continuous reliable provision of the service, even when the ISV is inaccessible because of a network failure.

### 3.5 Update of distributed metadata

One of the most important issues in a network such as the one described above is keeping up-to-date all the metadata it hosts. There is a need to develop a mechanism to publish possible changes on the proxylets provided by ISVs. When a new proxylet appears in an ISV site, or an existing proxylet is modified, the ISV announces this change to the PB it had initially contacted to publish his services. The ISV sends the new XML file to the PB, who in turn distributes this XML file to all nodes of the PDN. Upon receiving the new XML file, each PB updates its structures where proxylet associated information is stored. It may also be the case that the previous version of the software component represented by the replaced metadata has been downloaded recently. Thus, in order for the new metadata to be compatible with the corresponding Jar file, the PB should re-fetch the proxylet from the appropriate ISV.

## 4 Performance evaluation

In order to quantify the benefits obtained by the proposed PDN approach, we measure the performance of two experimental network topologies. In both topologies a client is assumed to issue requests for locating and fetching certain proxylets. A PDN infrastructure is available only in the second network topology

### 4.1 The network topologies

Both network topologies consist of 4 web servers, which act as ISVs, and are located at the addresses www.di.uoa.gr (ISV 1), theseas.softlab.ntua.gr (ISV 2), www.glue.umd.edu (ISV 3) and abs.telecom.ece.ntua.gr (ISV 4). ISV 3 is located in the U.S., while the other ones are located in Greece. Each ISV provides 7 proxylets (Jar files) of different sizes, so that there are, in total, 28 available proxylets. The selected sizes are 20, 50, 100, 250, 500 KB, 1 MB and 1.5 MB. Each ISV also provides 7 XML files, whose size is fixed at 6 KB, each one containing the metadata describing the corresponding proxylet. The client requesting the proxylets is located in the domain ntua.gr (same domain as ISVs 2 and 4). Figures 1 and 2 depict the two experimental topologies.

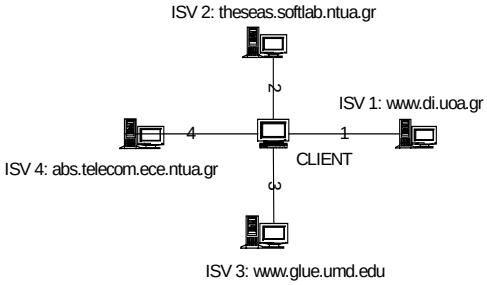


Fig. 1: Non-PDN topology

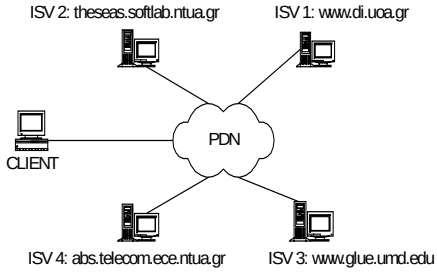


Fig. 2: PDN topology

### 4.2 Service establishment in the non-PDN topology

In this topology, the client maintains a list of ISVs providing proxylets, and queries them in order to locate the desired proxylet. The proxylet search is carried out with the help of metadata, which include the address of the ISV where it is located. Initially, the client selects an ISV from his list, either based on some policy, or in random, or in sequence. In our experiment the client scans the ISVs sequentially, in the order ISV 1, 2, 3, 4. Starting from ISV 1, the client issues 7 HTTP requests to get all metadata (XML files) available from ISV 1. After checking the metadata, if he finds the desired proxylet, he issues one more HTTP request to ISV 1, to retrieve the proxylet code (Jar file). In case the client does not find the desired proxylet in ISV 1, he proceeds with ISV 2, if he fails again he contacts ISV 3, and finally, if necessary, ISV 4. We measure the total response time, defined as the time that elapses from the moment the client issues the requests for the XML files of ISV 1, until the moment the desired Jar file is downloaded. We repeat our measurements for all cases where the desired proxylet is each one of the 28 proxylets provided by all ISVs.

### 4.3 Service establishment in the PDN topology

In the PDN topology, the client does not search for proxylets by connecting to ISVs directly, but instead relies on the functionality provided by the PDN. The client contacts the PB, who resides in the same domain (ntua.gr) as the client, and hosts all metadata from all ISVs (28 XML files). The connection between the client and the PB is based on socket communication; hence it is faster than HTTP. After receiving and parsing the XML metadata, the client requests that the desired proxylet be fetched. The PB subsequently makes one HTTP request to the appropriate ISV providing the proxylet, retrieves the Jar file and sends it back to the client, again through a socket connection. The PB also caches the downloaded Jar file. As in the previous experiment, we measure the resulting total response time until the Jar file is downloaded. We also measure the response time for the situation where the requested proxylet is already cached in the PB, in which case the ISV does not need to be contacted at all. This process is repeated for all 28 available proxylets.

### 4.4 Experimental results and discussion

We collect the response times obtained from the experiments described above in Fig. 3. The top left plot shows the response times for proxylets located in ISV 1, vs. the proxylet file size, the top right plot corresponds to proxylets located in ISV 2, etc.

The three lines in each plot represent the response times for downloading a proxylet, under the following three scenarios:

a) Non-PDN topology: The client sequentially scans ISVs 1 through 4 using HTTP, in order to find and download the appropriate proxylet. (Scenario labelled as Direct-ISV, in Fig. 3).

b) PDN topology: The client requests the proxylet via a PB participating in the PDN, who, in turn, fetches the proxylet from the appropriate ISV. (Scenario labelled as PB-ISV).

c) PDN topology: The client requests a proxylet that has already been cached by the PB. (Scenario labelled as PB-cache).

We observe that if the desired proxylet is found in either ISV 1 or ISV 2, the response times for scenario Direct-ISV are only slightly lower than those of scenario PB-ISV. That is, the resulting PDN performance is very much comparable even with the case where the client would need to

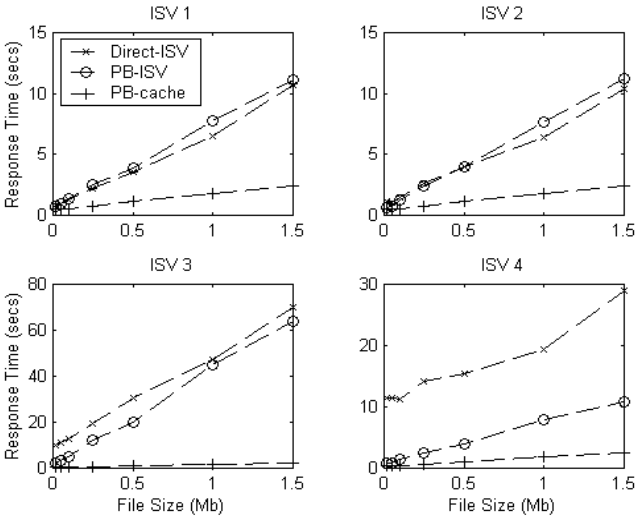


Fig. 3. (ISV 1) www.di.uoa.gr, (ISV 2) theseas.softlab.ntua.gr, (ISV 3) www.glue.umd.edu, (ISV 4) abs.telecom.ece.ntua.gr

directly query only one or two ISVs in order to retrieve the proxylet. Thus, the top two plots indicate that the overhead associated with the use of the PDN is rather small. Next, for the proxylets located in ISV 3, in which case scenario ISV requires sequential scanning of ISVs 1, 2, and 3, we see that the use of the PDN already leads to smaller response times than those of scenario ISV. This is due to the fact that scenario PB-ISV requires only one HTTP connection from the PB to the ISV where the proxylet is located, regardless of the number of ISVs, as opposed to multiple HTTP requests that may be necessary without the PDN. Obviously, as the number of uccessfully contacted ISVs increases, the response time under the Direct-ISV scenario also increases, and significantly exceeds that of PB-ISV. This difference between the response times of the Direct-ISV and PB-ISV scenarios becomes more pronounced when the proxylet is located in ISV 4, in which case the client needs to contact all four ISVs if no PDN is available. It is also worth noting that the smallest response times are achieved when the desired proxylet is found at the PB cache, as shown by the PB-cache lines in all four plots. The % response time improvement under the PB-ISV scenario, as compared to the Direct-ISV scenario, is summarised in Table 1.

| ISV 1 | ISV 2 | ISV 3 | ISV 4 |
|---|---|---|---|
| -8.43 % | 2.82 % | 20.21 % | 74.91 % |

Table 1. Response time reduction achieved by PDN

A further benefit of the PDN approach is that finding the desired proxylet in the PB cache yields a 70,71% response time reduction over the case where the proxylet is fetched using HTTP from ISV 1, i.e., the first ISV contacted by the client under scenario ISV.

In short, the PDN approach offers better performance than direct HTTP when the existing ISVs are three or more. Clearly, the performance gains of the PDN would become far more apparent in an active network with a large number of ISVs. This state of affairs should be typical of a realistic environment, where an increasing number of vendors provide their own proxylets to support add-ons or pure new network services.

## 5 Conclusions

We discussed an application level active network's main components and key characteristics. The need for a content distribution mechanism to spread out the services available on the network was documented, and our effort to implement a dedicated network of servers to provide part of a CDN's functionality was presented.

Experiments with the proposed content distribution architecture indicate that it may well offer a promising solution to the problem of distribution of services in an active networking environment. Information on services and the services themselves are distributed across a network of nodes placed close to users and the active servers, where they can be easily accessed. Reduced network traffic, quick response to applications that use those services and caching of those services are some of the benefits gained by this approach.

## 6 References


[1] E. Amir, S. McCanne and R. Katz, "An Active Service Framework and its Application to Real Time Multimedia Transcoding," *Proc. SIGCOMM'98*, pp. 178-189, September 1998.

[2] K. Calvert, S. Bhattacharjee, E. Zegura and J. Sterbenz, "Directions in Active Networks" *IEEE Communications Magazine*, 1998.

[3] M. Day, B. Cain and G. Tomlinson, "A Model For CDN Peering," IETF Internet Draft, http://www.alternic.org/drafts/drafts-d-e/draft-day-cdnp-model-01.html.

[4] M. Day and D. Gilletti, "Content Distribution Network Peering Scenarios," IETF Internet Draft, http://www.alternic.org/drafts/drafts-d-e/draft-day-cdnp-scenarios-00,01,02.html.

[5] D5: Active Networking Architecture, public document on http://www.cs.ucl.ac.uk/research/android/

[6] M. Fry and A. Ghosh, "Application Level Active Networking," *Fourth International Workshop on High Performance Protocol Architectures (HIPPARCH '98)*, June 1998.

[7] M. Fry and A. Ghosh, "Application Layer Active Networking," *Computer Networks*, Vol. 31, No 7, pp. 655-667, 1999.

[8] U. Legedza, D. J. Wetherall, and J. Guttag, "Improving the Performance of Distributed Applications Using Active Networks", *Proc. IEEE INFOCOM '98*, San Francisco (CA), USA.

[9] G. MacLarty and M. Fry, "Policy-based Content Delivery: An Active Network Approach," *Fifth International Web Caching and Content Delivery Workshop*, Lisbon, Portugal, 22-24, May 2000.

[10] I. W. Marshall and M. Banfield, "An Architecture For Application Layer Active Networking," *IEEE Workshop on Application Level Active Networks: Techniques and Deployment*, November 2000.

[11] I. W. Marshall, J. Crowcroft, M. Fry, A. Ghosh, D. Hutchison, D. J. Parish, I. W. Phillips, N. G. Pryce, M. Sloman and D. Waddington, "Application-Level Programmable Internetwork Environment," *BT Technology Journal* Vol. 17, No. 2, April 1999.

[12] D. Tennenhouse, J. Smith, D. Sincoskie, D. Wetherall and G. Minden, "A Survey of Active Network Research," *IEEE Communications Magazine*, Vol. 35, No. 1, pp. 80-86, January 1997.

[13] D. Tennenhouse and D. Wetherall, "Towards an Active Network Architecture," *Computer Communication Review*, Vol. 26, No. 2, pp. 5-18, April 1996.

[14] D. Wetherall, U. Legedza and J. Guttag, "Introducing New Internet Services: Why and How," *IEEE Network Magazine Special Issue on Active and Programmable Networks*, July 1998.