Heavy and Lightweight Dynamic Network Services : Challenges and Experiments for Designing Intelligent Solutions in Evolvable Next Generation Networks

Laurent Lefèvre
INRIA /LIP (UMR CNRS, INRIA, ENS, UCB)
Ecole Normale Supérieure de Lyon
46 allée d'Italie - 69364 Lyon Cedex 07 - France
laurent.lefevre@inria.fr

Abstract

Programmable and active networks allow specified classes of users to deploy dynamic network services adapted to data streams requirements. Based on our experience in high performance active networking, this paper compares two alternative approaches for adding dynamic solutions in the network : heavy and lightweight dynamic network services. We propose and describe solutions to efficiently manage and deploy heavy services (requiring resources and closely linked with middleware or applications) and lightweight network services (generic and pragmatic solutions with limited impact on network infrastructure). Experiments on local and wide area platforms are presented.

1. Introduction

Next generation evolvable networks will require flexibility for the deployment of network services. Programmable and active networks propose an alternative solution for network and protocols designers to dynamically deploy personalized services inside equipments. Active networks transform the *route-route* paradigm of IP networks to *route-process-route* abstraction. But how to design and evaluate the cost and impact of processing requirements to be able to support large scale deployment of services ?

We have recently shown that high performance software active networks environments can support Gbits networks[4, 5]. Most of experiments are performed on medium scale managed platform. We need now to find scalable solutions for a wide deployment of dynamic services on Internet. Two main kinds of services show different benefits. Heavy Services (requiring CPU resources) allow the deployment of high level functions in the network that can greatly support some applications (like Grid data streams [6]). On the opposite side, Lightweight Network Services (functions requiring a small specified amount of resources (CPU, states...)) seem a more pragmatic and generic way to widely deploy personalized services.

Our main problem is that the running duration of a service is unknown (depending on data stream length) and the resources needed by a new service will be only know during execution step. Moreover, previous works [3] have demonstrated that predictability of services consumption is not usable within programmable networks domain. So we must propose dynamic solutions to support heterogeneous active services. We present the *Feedback stream based* (FBSb) load balancing policy in order to efficiently deploy active services between internal nodes of a cluster-based active router. Based on a daemon collector and distributed agents, the policy improves load decisions taken by the programmable network equipment.

On the other way, proposing lightweight network services allow to easily design enough provisioned network equipments. But mapping and urbanizing these functionalities remains an open problem.

We base our deployment on the Tamanoir high performance execution environment[5]. We show that Tamanoir is able to support both kind of dynamic services on Gbit networks. We will show experiments done with Tamanoir deploying heavy services and lightweight network functionalities.

This paper shows that both approaches (heavy vs. lightweight services) are mandatory to dynamically support large classes of applications and that they open different problems.

The paper is organized as follows. In section 2, the Tamanoir active network environment is briefly described. We focus section 3 on how to support heterogeneous heavy network services. Section 4 presents architecture and first

experiments on lightweight network functionalities. We finish by some conclusions and future directions for the deployment of intelligent solutions in evolvable next generation networks.

2. Tamanoir : High Performance Execution environment

The integration of new and dynamic technologies into the shared network infrastructure is a challenging task, and the growing interest in the active networking field[7] might be seen as a natural consequence.

In our active networking vision, routers and any network equipments (like gateways, proxies,...) can perform computations on user data in transit, and end users can modify the behavior of the network by supplying programs, called *services*, that perform these computations. These routers are called *active nodes* (or *active routers*), and propose a greater flexibility towards the deployment of new functionalities, more adapted to the architecture, the users and the service providers requirements.

The aims of the Tamanoir[5] project is to design an high performance active node based on standard hardware and software and able to deploy services inside the network. Tamanoir Active Nodes (TAN) (Fig. 1) provide persistent active routers which are able to handle various data stream (audio, video, Grid...) on several planes (data,control and management) at the same time (multi-threaded approach). The both main transport protocol (TCP/UDP) are supported by the TAN for carrying data. We use the ANEP (Active Network Encapsulated Protocol)[1] format to send data over the active network.

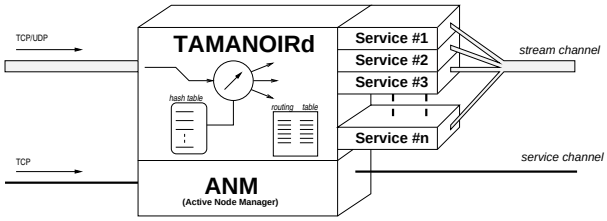

Figure 1. TAN : Tamanoir Active Node

Tamanoir supports these 4 different execution environment levels (Fig. 2) (programmable network interface card, kernel space, user space and distributed resources) in order to fit service requirement as smoothly as possible (in terms of CPU and memory requirements). We classify services in three kinds :

- lightweight service : stateless services or services requiring few processing capabilities;

Figure 2. TAN Architecture

- medium services : services requiring state and few computing and memory resources;
- heavy service : services applying intensive computation on the stream, or caching huge quantity of data.

3. Heavy network services

3.1. Supporting heterogeneous streams and services

High level and application oriented active services (compression, cryptography, transcoding on-the-fly...) require intensive computing resources. To support these heavy services, a Tamanoir Active Node embeds a dedicated cluster to efficiently deploy parallel services on streams. The main problem is that the running duration of a service is unknown (depending on data stream length) and the resources needed by a new service will be only know during execution step. Moreover, previous works [3] have demonstrated that predictability of services consumption is really not usable within programmable networks domain. So we must propose dynamic solutions to support heterogenous active services. We propose the *Feedback stream based* (FBSb) load balancing policy in order to efficiently deploy active services between internal nodes of the cluster.

We adapt the Linux Virtual Server (LVS)[8] software suite, dedicated to provide distributed servers like *ftp*, *web*, *mail*.... A Linux Virtual Server is a group of internal back-end nodes and a front-end.

We modify and adapt LVS functionalities for active networking and use it to distribute Tamanoir Execution Environment. A dedicated machine is configured as a front-end and is used to route packets from the Internet to back-end

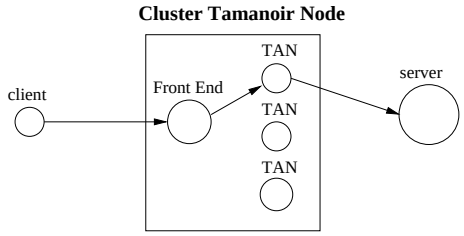

Figure 3. Tamanoir active router based on a cluster

Tamanoir nodes replicated on each machine of internal cluster (figure 3).

3.2. Feedback stream based load balancing policy (FBSb)

Common load balancing strategies (like Round Robin where the front end sequentially choose a back-end when a new stream crosses the cluster based equipment or Least Connected where the front-end chooses a back-end with the smallest number of active connections) can be efficient when deployed services are homogeneous (web or ftp server). For a cluster based active node, dynamic strategies must be deployed to ensure that back-end nodes are equally loaded with active services.

We focus on the front end machine which can take dynamic decisions when a new data streams arrive on the equipment. We add a weight table in the Front end in order to maintain a ”global” load view of the cluster. CPU and memory load are periodically transmitted to front end machine.

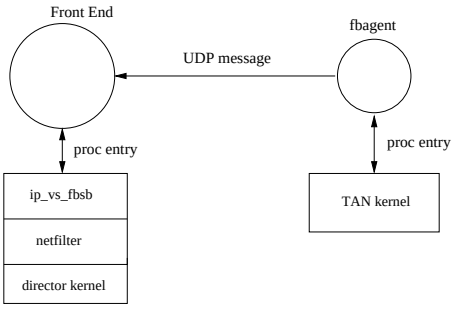

Figure 4. Architecture of FBSb policy

The FBSb architecture is based on a central collector and distributed agents :

3.2.1 FELC : Front End Load Collector

The FELC is the daemon running on the LVS font-end machine in order to collect information about load from each agent. The other task of FELC is to provide this information to the kernel space (trough a *proc* entry).

The IP Virtual Server (*ipvs*) FBSb scheduling module supports three tasks. The fist one is to set up the *proc* entry where FELC will write the load of the nodes. The second one is to choose a node (the less loaded). The last one is to remove the proc entry when unloaded. When a new data stream is sent to the ”best” back end, the front end machine virtually increases the workload of the back-end in its table in order to avoid re choosing the same machine when a new stream arrives.

3.2.2 FeedBack Agent

An agent (*fbagent*) is deployed on all back-end nodes deploying Tamanoir. This component monitors the CPU load of the back-end node (Figure 4) and periodically reports this *CPU load* to the Front End Load Collector. The transaction is done over UDP protocol. In order to reduce the traffic fbagent provides caching capabilities in order to only send the *CPU load* when this measure goes over a threshold.

3.3. Experiments

Our experiments have been made on a local platform of 4 Compaq DL360 Bi-Pentium III 1.7 Ghz for the cluster Tamanoir node and 12 SUN LX50 for client machines connected through Myrinet[2] network.

We validate our cluster based active router and associated policies with a wide set a scenario. Few of them are presented here. We select two different kind of services:

- Heavy service (3-DES encryption) : this service makes an intensive use of CPU (1 stream calling this service saturates a CPU, in our test implementation);
- Medium service (stateful traffic analysis) : this service uses the CPU in a less intensive way, but its impact is not transparent (3-4 streams calling this service saturate a CPU, in our test implementation).

We validate two kind of services deployment :

- *200*: composed by 1 stream calling a service of encryption and 11 streams calling for a traffic analysis with different delays of arrival of 1 second (*201*), 2 seconds (*202*) and 5 seconds (*205*).
- *300*: composed by 4 streams calling a service of encryption and 8 streams calling for traffic analysis. with different delays of arrival of 0 second (*300*), 1 second (*301*) and 2 seconds (*302*).

Table 1. Feedback stream-based, Round Robin and Least Connected load balancing policies

	FBSb			RR			LC		
TEST	MAX	AVG	MIN	MAX	AVG	MIN	MAX	AVG	MIN
201	38.22	20.20	15.98	43.25	20.13	15.51	38.50	20.31	15.43
202	33.31	20.04	15.73	37.49	20.12	15.41	36.44	20.17	15.47
205	32.92	19.42	15.53	38.81	19.62	15.45	37.38	19.61	15.44
300	52.68	24.16	16.15	93.64	26.06	15.45	64.65	25.50	16.36
301	50.83	24.63	15.47	90.31	26.93	16.12	64.01	25.60	15.40
302	50.96	23.29	15.38	92.90	25.53	15.92	57.83	25.07	15.68

We compare our Feedback stream based policy with Round Robin (RR) and Least Connected (LC) policies (table 3.2). We can note here that RR and LC strategies provide efficient results to the 20x test case due to the nearly homogeneous scenario (only one heavy service and 11 medium services). When services distribution becomes strongly heterogeneous (case 30x), we can note that a feedback based policy clearly improves performances of the whole equipment. We also observe that FBSb policy strongly reduces deviation of experiments (table 3.3).

Table 2. Mean and standard deviation for 3 load balancing policies

		FBSb		RR		LC
TEST	SDeV	AVG	SDeV	AVG	SDeV	AVG
201	3.31	20.20	4.30	20.13	3.75	20.31
202	3.09	20.04	3.93	20.12	3.58	20.17
205	3.26	19.42	3.97	19.62	3.81	19.61
300	9.12	24.16	14.96	26.06	12.00	25.50
301	8.87	24.63	14.87	26.93	10.43	25.60
302	8.78	23.29	13.61	25.53	10.52	25.07

We have briefly presented our architecture for the deployment of a Feedback stream based load balancing strategy. First experiments show needs and benefits of having such policy when heavy heterogeneous services are deployed in a cluster-based active equipment.

4. LNF : Lightweight Network Functionalities

4.1. Mapping LNF

A lot of deployed active and programmable network solutions are performed on controlled platforms. It remains hard to evaluate the need of computing resources in cluster based active equipments to support large number of heavy services. We need to associate heavy services with pragmatic and generic dynamic functionalities which could support scalability and tolerance issues.

Lightweight network services seem a more pragmatic way to support large number of heterogeneous streams. A Lightweight Network Functionality (LNF) is a network service requiring a small specified amount of resources (CPU, states...).

Main problem concerns the mapping and urbanization of network functionalities. How to imagine a large deployment of dynamic and programmable network services on Internet and how to map services on network equipments ? We can classify different mapping scenario depending on openness of the network :

- Service mapped on the data path
  - A service (eventually composed on several blocks) is deployed on a specific dedicated equipment located on the data path (Fig. 5).

Figure 5. Service on data path

  - Service replicated on several network equipments (Fig. 6). This approach can be linked with active network solutions.

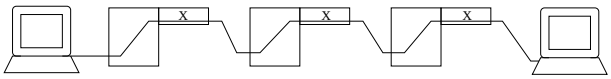

Figure 6. Replicated service

  - Service distributed on various equipments (Fig. 7). This approach allows to support pipelined services but needs to deal with fault tolerance (if one of the service block crashes).
- Services outside the data path
  - Service mapped out of the data path : allows to map a service on a dedicated equipment (legacy solution) (Fig. 8).

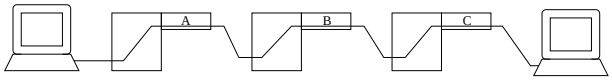

Figure 7. Distributed service

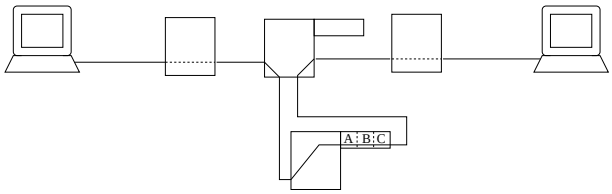

Figure 8. Unique service out of the data path

– Distributed service outside the data path : require monitoring sensors to evaluate the cost of forwarding streams several times during transport. (Fig. 9)

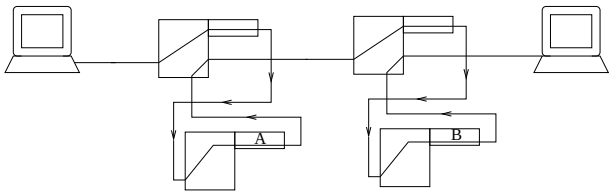

Figure 9. Distributed service outside the data path

4.2. Supporting LNF in Tamanoir Active Nodes

Within software based active routers, operating systems play an important role. Recent version of Linux provide possibilities to support filtering functionalities in the kernel (NetFilter module). With *hooks* (fig. 10) linked to specific packet actions, users can run personalized applications.

The Tamanoir system is an active Execution Environment which allows users to efficiently deploy personalized services inside the network. We propose to deport lightweight active services from distributed resources and user space (high level JVM) to kernel space (low level NetFilter modules). Our goal is to provide new levels of performance to software active routers.

The various modules which are set up into the OS kernel can be modified dynamically by active services. A Tamanoir active service, running inside the user space configure the LNF in kernel by sending a control message (see Fig. 11) to the LNF. This on-the-fly configuration allows to dynamically deport personalized function inside the kernel.

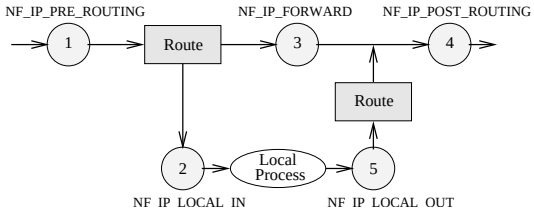

Figure 10. NetFilter Hooks

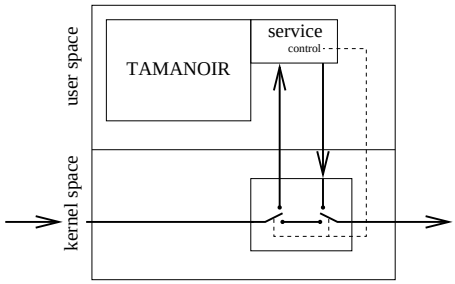

Figure 11. Mixing LNF in kernel and heavy service in user space

4.3. Experiments

We experiment the deport of active forwarding service inside the kernel for UDP active packets (Fig. 12).

Packets crossing the Tamanoir active node remaining in the Linux kernel layer spend around 7 microseconds for basic forwarding operations with UDP (Fig. 12). This is small compared to user space version with with standard Java Virtual Machines (SUN, IBM or compiled version (GCJ))

These experiments of deported services show the flexibility for configuring the NetFilter module from Tamanoir Service. By using standard messages, we can easily configure and active NetFilter module to dynamically support Lightweight Network functionalities in the kernel of active nodes.

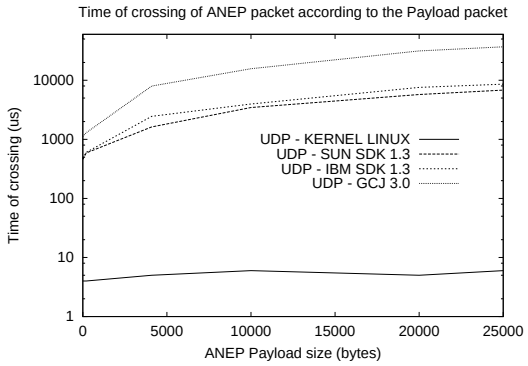

Figure 12. ANEP packets using UDP protocol processed by heavy services (in JVM / user space) or LNF (in kernel space)

5. Conclusion and future works

Next generation evolvable networks will require flexibility in terms of network services. We have shown that Tamanoir project is able to support current bandwidth network requirements (Gbits) with the help of optimized architecture. But heterogeneous streams and services require efficient solutions for scalable architecture and networks.

Both approaches (heavy vs. lightweight services) require various support and respond to various requirements from applications. Mixing both of them (on network architecture or inside a node) seems also to be a performant solution for large networks.

We want to more deeply explore the heterogeneous aspects of cluster based programmable nodes by taking into account of heterogenous clusters (in terms of resources (CPU, memory) and dedicated hardware (cryptographic card, video card...)) in order to adapt dynamically our load balancing policy.

Our current activities concern the deployment and validation of LNF at large scale in an emulated large platform (RNRT Grid5000 project).

Acknowledgements

The author wish to thank P. Giacomin and J.P. Gelas for their help during evaluation step.

References

[1] S. Alexander, B. Braden, C. Gunter, A. Jackson, A. Keromytis, G. Minden, and D. Wetherall. Active Network Encapsulation Protocol (ANEP). RFC Draft, Category : Experimental, July 1997.

[2] N. Boden, D. Cohen, R. Felderman, A. Kulawik, C. Seitz, J. Seizovic, and W.-K. Su. Myrinet : a gigabit per second local area network. *IEEE-Micro*, 15(1):29–36, Feb. 1995.

[3] V. Galtier, K. Mills, and Y. Carlinet. Modeling cpu demand in heterogeneous active networks. In *2002 DARPA Active Networks Conference and Exposition (DANCE'02)*, San Francisco, CA, May 2002.

[4] J.-P. Gelas, S. El Hadri, and L. Lefèvre. Tamanoir: a software active node supporting gigabit networks. In *ANTA 2003 : The second International Workshop on Active Networks Technologies and Applications*, pages 159–168, Osaka, Japan, may 2003.

[5] J.-P. Gelas, S. E. Hadri, and L. Lefèvre. Towards the design of an high performance active node. *issue of Parallel Processing Letters (PPL) journal*, 13(2):149–167, June 2003.

[6] L. Lefèvre, C. Pham, P. Primet, B. Tourancheau, B. Gaidioz, J. Gelas, and M. Maimour. Active networking support for the grid. In N. W. Ian W. Marshall, Scott Nettles, editor, *IFIP-TC6 Third International Working Conference on Active Networks, IWAN 2001*, volume 2207 of *Lecture Notes in Computer Science*, pages 16–33, Oct. 2001. ISBN: 3-540-42678-7.

[7] D. Tennenhouse and D. Wetherall. Towards an active network architecture. *Computer Communications Review*, 26(2):5–18, April 1996.

[8] W. Zhang. Linux Virtual Server for Scalable Network Services. In *Ottawa Linux Symposium*, 2000. http://www.linuxvirtualserver.org.