TAMANOIR : A HIGH PERFORMANCE ACTIVE NETWORK FRAMEWORK

Jean-Patrick Gelas and Laurent Lefèvre
RESAM Laboratory - Université Claude Bernard Lyon1
Batiment ISTIL, 43 bd du 11 Novembre 1918
69622 Villeurbanne - France
jpgelas@resam.univ-lyon1.fr, llefevre@resam.univ-lyon1.fr

Keywords: Active networks, performance, Tamanoir, mobile code, ANTS, Java

Abstract The field of active and programmable networks is rapidly expanding. These networks allow users and network designers to easily deploy new services. While most of proposed systems deal with adaptability, flexibility and new protocols only few systems focus on high performance.

We explore the design of intelligent network by proposing a new kind of architecture dedicated to high performance active networking.

This paper presents the Tamanoir[1] system, a complete framework that allows users to easily deploy and maintain distributed active routers on wide area networks. A set of distributed tools is provided : routing manager, active nodes and stream monitoring tool, web-based services library... Based on JAVA/GCJ with multi-threading approach to combine performance and portability of services, applications can easily benefit of personalized network services. First experiments and comparisons with other systems will show the effectiveness and pertinence of our approach.

[1] Tamanoir (*great anteater*) is one of the strangest animal of south America : living in savanna, with an impressive tongue with a mouth of 2 centimeters diameter this animal only eats ants (30000 daily). We choose this animal in reference to ANTS system.

Introduction

The field of active and programmable networks is rapidly expanding. These networks allow users and network designers to easily deploy new services. While most of proposed systems deal with adaptability, flexibility and new protocols only few systems focus on high performance.

We explore the design of intelligent network by proposing a new kind of architecture dedicated to high performance active networking. Tamanoir framework is based on an architecture where active routers are only deployed on network periphery while backbones remain passive to guarantee performances. Active services can be easily deployed in the network and are adapted to architecture, users and service providers requirements. High performance aspects are also taken into account by minimizing JAVA software latency with compilation approach. Our system provides persistent active routers which manage multi-applications and various data streams (audio, video...). The Tamanoir system is a complete framework that allows users to easily deploy and maintain distributed active routers on wide area networks. A set of distributed tools is provided : routing manager, active nodes and stream monitoring, web-based services library... Based on compiled JAVA/GCJ with multithreading approach to combine performance and portability of services, applications can easily benefit of personalized network services. Experiments and comparisons with other active network systems will show the effectiveness and pertinence of our approach.

This paper reports on our experience in implementing Tamanoir (section 1) and designing a complete framework for active networks deployment (section 2). In section 3, we compare our work with other related projects and experiment Tamanoir in section 4 . We conclude in section 5 and present our future works.

1. TAMANOIR ARCHITECTURE

1.1. ACTIVE ARCHITECTURE

Tamanoir active architecture is based on a virtual topology of active nodes spread on programmable routers of the network. Tamanoir active routers, called Active Nodes (AN), are deployed on network periphery. Contrary to others approaches, we do not believe in the actual deployment of Gigabit active routers for backbones. If we consider that the future of WAN backbones could be based on all-optical networks, no dynamic services will be allow to process data packets. So, we prefer to

consider backbones like high performance well-sized passive networks. We only concentrate active operations on routers/nodes at network periphery.

We implement applications typically composed of senders and receivers which exchange data streams (video, audio...). Data streams cross various active nodes up to passive backbone and then cross another set of active nodes up to receiver node (see figure 1). Tamanoir architecture is based on Active Node approach : programs, called services, are injected into active nodes independently of data stream. Active nodes apply these services to process data streams packets. Services are deployed on demand when streams arrive on an active node.

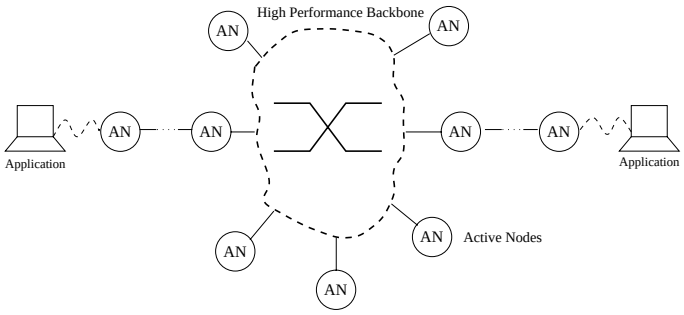

Figure 1 Heterogeneous Active Architecture with high performance passive backbone and active nodes on periphery

1.2. CAPSULES

Tamanoir capsules are based on Active Network Encapsulation Protocol (ANEP) format defined by The Active Network Group [Alexander et al.,]. Each ANEP packet is encapsulated in an UDP datagram. A Marshalling operation is needed before each packet send or receive operation. An ANEP packet is based on an header containing TLVs (Type, length, Value) entities. These TLVs are decoded and interpreted by active nodes to apply dynamic services on the packet. Most important TLVs are fixed : name of sending machine, name of receiving machine... Tamanoir capsules use personalized TLVs like : service name, last crossed active node...

1.3. OVERVIEW OF A TAMANOIR NODE

An active node is a router which can receive capsules of data, process them and forward them to other active nodes.

A Tamanoir Active Node (TAN) is a persistent daemon acting like a dynamic programmable router. Once deployed on a node, it is linked

to its neighbors in active architecture. A TAN receives and sends capsules of data after processing them with user services. A TAN is also in charge of deploying and applying services on capsules depending on application requirements. When arriving in Tamanoir daemon, a capsule is forwarded to service manager (figure 2). The capsule is processed by service in a dedicated thread. Resulting capsule is then forwarded to next active node or to receiver part of application according to routing tables maintained in TAN.

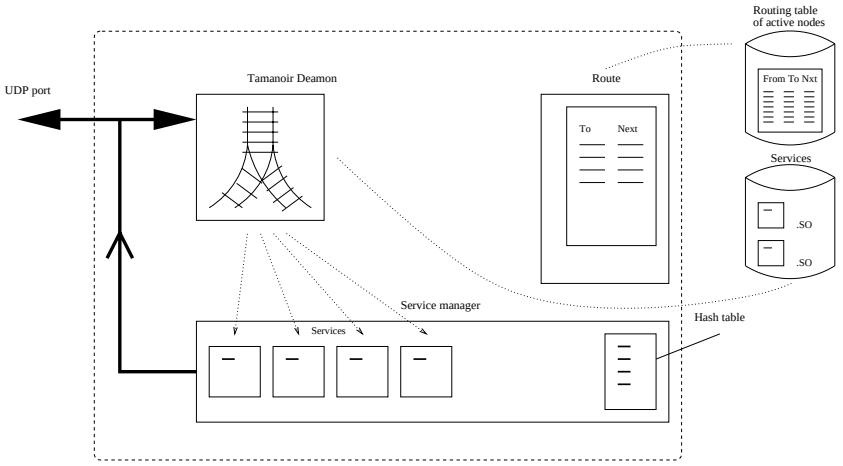

Figure 2 Tamanoir Active Node

1.4. SERVICE DEPLOYMENT

A Tamanoir service is a JAVA class (see Figure 3) containing generic methods like :

- receive : used to specify the code applied to capsules received by Tamanoir node;
- send : used to forward capsules to another active node or to the receiving application;
- start/stop : useful for service initializing and service stop.

Each new service is directly derived from the generic service class. This operation allows to dynamically load a service into TAN memory. Each new service is deployed TAN after TAN, when the first capsule which needs a new service is processed by the node. So the reduced cost to deploy a new service consists of asking this new service to the last active node crossed by the capsule. By this way, we limit the impact of new service deployment.

Figure 3 Generic service class

```
public class Service {

  UDPnetworkTools udp;

  public Service() {
    udp = new UDPnetworkTools();  }

  public void recv( String srcId, destId, lastId, byte [] payload ) {}
  public void send( String srcId, destId, lastId, byte [] payload ) {}
  public void start() {}
  public void stop()  {}
  public void process( byte [] payload ) {}
} // class Service
```

2. TAMANOIR FRAMEWORK

2.1. ROUTING DAEMONS

A Tamanoir topology can be dynamically built by the user. Active nodes can be add or remove during applications runs. Routing daemons are linked to their neighbors and they propagate and update routing tables in a non centralized server architecture.

2.2. SERVICES LIBRARY

We are also currently designing a web service broker where applications can decide to use generic Tamanoir services. By using an *http address* in service name, applications deploy non-personalized services. These services are then downloaded from a web server and implemented on active nodes.

2.3. TAMT : TAMANOIR-ANTS MONITORING TOOL

Active networks systems allow users to easily deploy services adapted to their applications. In compensation, such systems often appear, from the programmer point of view, as a "black box" since no information about active architecture is available. Consequently, in the absence of visualization and monitoring tools, optimizing, debugging or evaluating the performance of applications is very difficult.

We are currently developing a monitoring tool, called TAMT (Tamanoir Ants Monitoring Tool), dedicated to active networks. This tool has been ported on top of Tamanoir and ANTS system.

Various kind of information are provided by TAMT (see figure 5) :

- active architecture : routing topology;

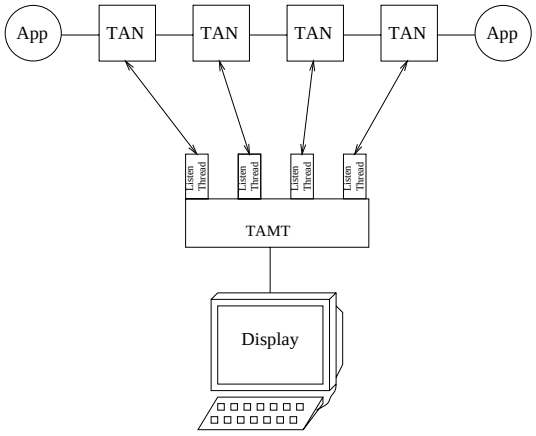

Figure 4 Tamanoir Ants Monitoring Tool architecture : the Visualization Process communicates with all active nodes involved in application

- active nodes information : address, name;
- capsule data streams : size, throughput, number of capsules

Monitoring features are embedded to ANTS or Tamanoir active node. Information about node and capsules are collected, compacted and sent periodically to TAMT server (see figure 4). Each active node is linked to a dedicated thread running on the visualization console.

Monitoring data can be visualized online during capsules transport. Off-line summarized views are also provided.

3. RELATED WORKS

Interest in active networks is really active since 1996 [Tennenhouse and Wetherall, 1996]. We have specially focus on two well-known active networks approaches : ANTS [Wetherall et al., 1998] and PAN [L.Nygren et al., 1999].

ANTS is one of the first and standard implementation of active architecture. Completely written in Java, ANTS nodes are executed on each router in a Java Virtual Machine. Focusing on security aspects, ANTS nodes check and validate each bytecode services before running on active nodes. Most of persistent and multi-applications aspects are not available and the intensive use of interpreted JAVA services limits high performance capsules transport and processing.

PAN is the first active network system dealing with performance issues. The development of PAN drew heavily of on many of the ideas

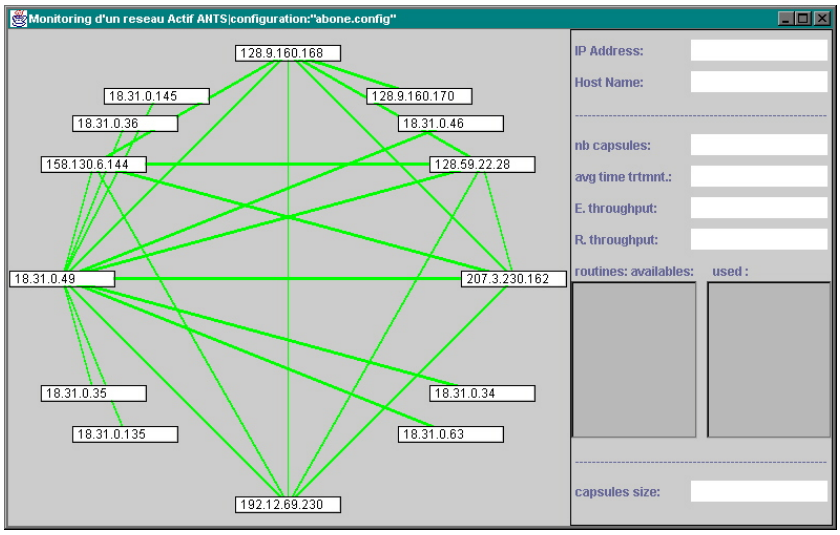

Figure 5 View of TAMT framework with routing topology of active nodes and monitoring information on nodes, capsules and services

developed in ANTS. PAN is completely written in C and provides good raw performances. Two implementations are available : in user-space or in kernel space. No portability and security issue are taken into account in PAN but all developments are dedicated to performance with intensive use of zero-copy protocols. A qualitative comparison of ANTS, PAN and Tamanoir is summarized in table 1.

Another approaches on high performance active networks are also currently explored like ANN [Decasper et al., 1999] and CANEs [Merugu et al., 1999] projects.

Table 1 Qualitative aspects of ANTS, PAN and TAMANOIR implementations

	ANTS	PAN	TAMANOIR
open source	yes	no	yes
portability	yes (Java)	no (C)	yes (Java+GCJ)
performance aspects	no	yes	yes
security	yes	no	no
dynamic service load	yes	yes	yes
multi-services	no	yes	yes
dynamic topology	no	yes	yes
implementation	3 years	3 years	5 months

4. EXPERIMENTS

We based our experiments on Pentium II 350 MHz linked with Fast Ethernet switches (100 Mb/s) and JDK 1.1.7 on Linux Debian distribution.

After first experiments of ANTS, this system really shows a lack of performance. Between ANTS choice of Java bytecode (portable and slow) and PAN choice of C native code (non portable and fast), we prefer to choose an hybrid approach. We combine Java language to provide portability to services and active nodes with compiled running to obtain performances. Our choice was influenced by the Jaguar [Welsh, 1999] environment which focus on high performance Java. We based Tamanoir implementation on GNU Compiler for Java (GCJ [GCJ,]), a recent Java code compiler generating native machine language.

4.1. COMBINING PERFORMANCE AND PORTABILITY

Our first experiments are based on a ping application, useful to check a distant active router. We use this application to compare Tamanoir running with JVM versus GCJ compiled version.

Experiments (in figure 6 and 7) show that implementing Tamanoir with GCJ increases performances by 2 times (300 ms needed to cross 3 active nodes in both way with GCJ while JVM version requires 600 ms).

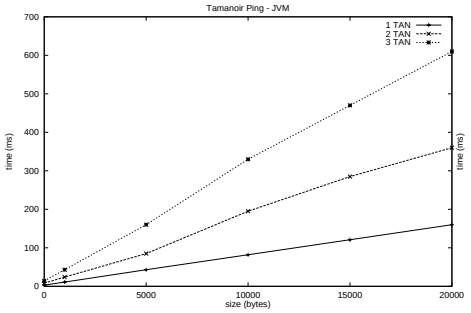

Figure 6 Ping Service with Tamanoir/JVM

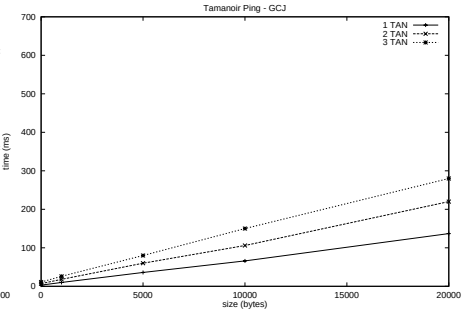

Figure 7 Ping Service with Tamanoir/GCJ

4.2. COMPARISON WITH ANTS

Performance comparisons with ANTS (version 1.2) have been made by using on a 3-nodes topology : one active node, one sender and one receiver node. We compare both systems by sending long capsule streams

trough active node. These capsules are processed by active node which receives them, applies *ForwardService* and forwards capsules to receiver node.

Results presented in figure 8 show the delay needed to cross an active node (latency). While ANTS needs 3 ms and is dependent of capsule payload size; Tamanoir time remains constant with a latency of 750 μs. Meanwhile, ANTS process capability is weak with an average number of 350 capsules/sec (figure 9). Tamanoir goes 3 times faster and processes up to 1350 capsules per second (depending of payload size).

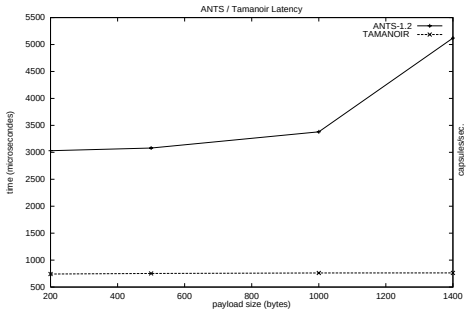

Figure 8 Latency : cost to cross an active node

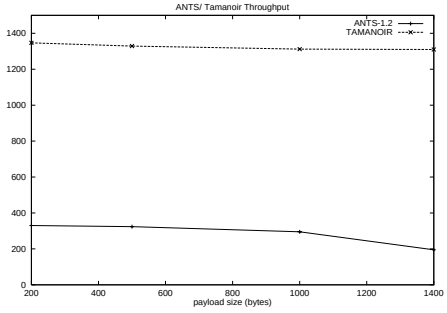

Figure 9 Number of capsules processed by an active node

No comparisons with PAN system were possible due to lack of system availability.

5. CONCLUSION AND FUTURE WORKS

By combining compiled implementation with Java portability; we try to develop a high performance portable active network system with multi-applications persistent routers and dynamic services. We also propose a complete framework to easily deploy, manage, evaluate and monitor active network architecture. Our Tamanoir framework is the first step to fully understand how high performance active networks can be designed.

Further experimentations are needed to fully understand the impact of JAVA/GCJ design on Tamanoir implementation. We are currently implementing new Tamanoir services for distributed QoS servers and optimized active multicast services. We would also like to benchmark our work with other high performance active networks framework like PAN, ANN and CANEs.

We will also go further in high performance active network designing by implementing an active node based on Myrinet [Boden et al., 1995]

cluster. More information on Tamanoir project can be obtained on $http://resam.univ-lyon1.fr/\tilde{}llefevre/TAMANOIR$

Acknowledgments

We would like to thank Matt Welsh for his help in understanding high performance Java with his Jaguar tool [Welsh, 1999]. We would also like to thank Arnaud Mergey for his help in designing Tamanoir Ants Monitoring Tool.

References

Alexander, D., Braden, B., A.Gunter, C., W.Jackson, A., D.Keromytis, A., and andDavid Wetherall, G. J. Active network encapsulation protocol (ANEP). RFC Draft, Category : Experimental, http://www.cis.upenn.edu/switchware/ANEP/.

Boden, N., Cohen, D., Felderman, R., Kulawik, A., Seitz, C., Seizovic, J., and Su, W.-K. (1995). Myrinet : a gigabit per second local area network. *IEEE-Micro*.

Decasper, D., Parulkar, G., Choi, S., DeHart, J., and Wolf, T.and Plattner, B. (1999). A scalable, high performance active network node. In *IEEE Network*, volume 13.

GCJ. The gnu compiler for the java programming language. http://sourceware.cygnus.com/java/.

L.Nygren, E., J.Garland, S., and Kaashoek, M. (1999). Pan: A high performance active network node supporting multiple mobile code systems. In *IEEE OPENARCH '99*.

Merugu, S., Bhattacharjee, S., Chae, Y., Sanders, M., Calvert, K., and Zegura, E. (1999). Bowman and canes: Implementation of an active network. In *37th Annual Allerton Conference, Monticello, IL*.

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

Welsh, M. (1999). A system supporting high-performance communication and i/o in java,. Master's thesis, University of California, Berkeley.

Wetherall, D., Guttag, J., and Tennenhouse, D. (1998). ANTS : a toolkit for building and dynamically deploying network protocols. In *IEEE OPENARCH '98*.