Tamanoir : a High Performance Active Network Framework

Jean-Patrick Gelas and Laurent Lefèvre
RESAM Laboratory - INRIA Action RESO

llefevre@resam.univ-lyon1.fr
http://resam.univ-lyon1.fr/TAMANOIR
What do we expect from networks?

Performance

- low latency, high bandwidth, low packet lost, fault tolerance...

Services

- QoS, multicast, DiffServ...

Elegance

- easy plug-and-play new services;
- dynamic deployment of new protocols.

But reality is really different!!!
What do we obtain from networks?

**Performance**

- traffic jam, reduced bandwidth

**Services**

- only few services available (best effort)...

**Elegance**

- static protocols...

An active approach?
Plan

1. Introduction
   - Active networks : various approaches
   - Existing environments : ANTS and PAN

2. A new model for High Performance Active Networks
   - Packets, services, active nodes.
   - Implementation strategies (language, compiler)
   - Tools box : TAMT

3. Experiments :
   - latency, bandwidth...
   - qualitative aspects

4. Conclusion and future works
Active networks

– active research field (since 1995);
– Goal: Internet use in a more intelligent way
– Intelligent switches (nodes) run applications (services) to process data stream;
– Dynamic services deployment.
Two approaches

Active packets:
- A service in each data packet

Active/programmable nodes:
- Services deployed concurrently to data one time
- Each packet contains a reference to a service present on the active node.
Now

Passive networks
- High performance Core network
- IP
- Long distance

Active networks
- Services
- Pseudo dynamic deployment
- Optimized traffic
Our approach

Link the two worlds

Réseaux passifs
- Core Network Hautes performances
- IP
- Longue distance

Réseaux actifs
- Services
- Personnalisation
- Trafic optimisé
Our approach

Link the two worlds

High performances Active Networks

✓ High performance stream process
✓ Long distance
✓ Dynamic services
Existing Environment (I)

ANTS [David J. Wetherall, Fév. 1999]

• written in JAVA;
• run in a JVM;
  – security, code validation…
  – interpreted bytecode (portability)
• complete model;
• limited implementation;
  – no multi-application;
  – no high performances.
Existing Environment (II)

PAN [Erik L.Nygren, Fév. 1998]

- high performance;
- written in C;
- two implementations (kernel, user-space);
- good raw performances;
- no security
A new model of High performance Active Network

- peripheral model;
- optimized active nodes;
- standard ANEP packets;
- multi-services;
- dynamic deployment;
- complete environment (tool box).
Active nodes on peripheral

High performance passive core network
High Performance Backbone

Active Nodes

Data streams
TAMANOIR Environment Implementation

- **JAVA Language**
  - high level language, standard libraries;
  - interpretation too slow.
- **Using GCJ : GNU Compiler for Java**
  - portable and optimized JAVA compiler;
  - native code execution.

<table>
<thead>
<tr>
<th>Java (javac)</th>
<th>C (gcc)</th>
<th>Java (gcj)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2459</td>
<td>125</td>
<td>176 (\times 13.9)</td>
</tr>
<tr>
<td>10.000.000 for loop (ms.)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Packets/capsules

<table>
<thead>
<tr>
<th>Version</th>
<th>Flags</th>
<th>Type Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header Length</td>
<td>Packet Length</td>
<td></td>
</tr>
</tbody>
</table>

Options

Payload

ANEP packets (*Active Network Encapsulation Protocol*) defined by *Active Network Group*
A TAMANOIR Active Node

- Daemon waiting UDP packets
- Few-copies protocols
- Multi-threading
Services

- A service is a specific application with generic methods `send()`, `recv()`
- Each service is derived from a generic class.

```java
public class Service {
    (...)
    public void recv( String srcId, destId, lastId, byte [] payload ) {}
    public void send( String srcId, destId, lastId, byte [] payload ) {}
    (...)
}

public class PingService extends Service {
    (...)
    public void recv( String srcId, String destId, String lastId, byte [] payload ) {
        if( localId.equals(destId) ) {
            send( destId, srcId, lastId, payload ); // come back to the sender
        } else {
            send( srcId, destId, lastId, payload ); // forward
        }
    }
    (...)
```
ANM
(Active Node Manager)

- dynamic service loading;
- dynamic topology: local synchronizations of AN.
TAMT : Tamanoir-Ants Monitoring Tool

TAMT visualization :
- active architecture : routing topology ;
- active nodes information : address, name ;
- capsule data streams : size, throughput, number of capsules
Experiments

- Ping Application -

![Graph showing Ping Tamanoir - 3 hop](image)

<table>
<thead>
<tr>
<th>TAMANNOIR</th>
<th>JVM</th>
<th>GCJ</th>
</tr>
</thead>
<tbody>
<tr>
<td>latency</td>
<td>1400 (\mu s)</td>
<td>754 (\mu s)</td>
</tr>
<tr>
<td>bandwidth</td>
<td>720 p/s</td>
<td>1324 p/s</td>
</tr>
</tbody>
</table>
Experiments: video stream with ANTS and Tamanoir

Latency

Bandwidth

ANTS / Tamanoir Latency

ANTS-1.2
TAMANOIR/GCJ
TAMANOIR/JAVA

ANTS / Tamanoir Throughput

ANTS-1.2
TAMANOIR/GCJ
TAMANOIR/JAVA

payload size (bytes)

payload size (bytes)
# Qualitative aspects of projects implementations

<table>
<thead>
<tr>
<th>Feature</th>
<th>ANTS</th>
<th>PAN</th>
<th>TAMANOIR</th>
</tr>
</thead>
<tbody>
<tr>
<td>open source</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>portability</td>
<td>yes (Java)</td>
<td>no (C)</td>
<td>yes (Java+GCJ)</td>
</tr>
<tr>
<td>performance aspects</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>security</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>dynamic service load</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>multi-services</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>dynamic topology</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>implementation</td>
<td>3 years</td>
<td>3 years</td>
<td>5 months</td>
</tr>
</tbody>
</table>
Conclusion

- Non all-active model: active nodes outside core network;
- Tamanoir: a complete framework (>3200 JAVA lignes);
  - optimized implementation: reduced copies protocols, multithreading...
  - dynamic services deployment;
  - multi-services and multi-applications support;
  - dynamic topology.
- Experimental validation;
Future works

• active node in kernel space;

• new services:
  – distributed QoS service;
  – multicast service.

• security;

• active node on high performance cluster.