Towards the design of an industrial autonomic network node

M. Chaudier, J.-P. Gelas, and L. Lefèvre

INRIA / LIP (UMR CNRS, INRIA, ENS, UCB 5668)
École Normale Supérieure de Lyon
46 allée d'Italie - 69364 Lyon Cedex 07 - France
mchaudier@ens-lyon.fr, jpgelas@ens-lyon.fr,
laurent.lefevre@inria.fr

Abstract. Programmable and active networks allow specified classes of users to deploy dynamic network services adapted to data streams requirements. Currently most of researches performed on active networks are conducted in research laboratories. In this paper, we explore the design of IAN^2 an **I**ndustrial **A**utonomic **N**etwork **N**ode able to be deployed in industrial context. Performance, dynamic programmability and fault-tolerance issues of software and hardware components have been prospected. First experimental evaluations on local platforms are presented.[1]

1 Introduction

Research works about active and programmable networks and evaluation of the experimental prototypes take place mostly in academical research laboratory. Currently no "plug and process" active equipments are available on the market place.

In the framework of a cooperative industrial maintenance and monitoring project (TEMIC project[3]), in which we are currently involved with different academic and industrial partners, we design devices to be easily and efficiently deployable in an industrial context. Once the hardware deployed and used, it must also be easily removable at the end of the maintenance or monitoring contract. In this project, we deploy our devices in secured industrial departments, restricted areas, or in an out-of-the-way locations. These devices must act as auto-configurable and re-programmable network nodes. Thus, the equipments must be *autonomic* and must not require direct human intervention.

The design of an autonomic network equipment must take into account specific requirements of active equipments in terms of dynamic service deployment, auto settings, self configuration, monitoring but also in terms of hardware specification (limited resources, limited mechanical parts constraints, dimension constraints), reliability and fault tolerance.

[1] This project is supported by french RNRT Temic [3] project with SWI Company, INRIA, GRTC and LIFC.

This paper presents our current work on the design and adaptation of an industrial autonomic network node. We propose an adaptation of a generic high performance active network environment (Tamanoir [8]) in order to deploy on limited resources based network boxes and to increase reliability and scalability. The implementation process is based on a hardware solution provided by the Bearstech[2] company. Through this approach we propose the architecture of an **I**ndustrial **A**utonomic **N**etwork **N**ode (called IAN^2) able to be deployed in industrial platforms. We evaluate the capabilities of IAN^2 in terms of computing and networking resources and dynamic re-programmability.

This paper is organized as follows. Hardware and software are respectively described in section 2 and 3. Section 4 shows first performance evaluation of the IAN^2. Section 5 briefly covers others works on industrial active nodes and finally the paper concludes in section 6.

2 Hardware platform

This section describes briefly the hardware used to implement the IAN^2 industrial autonomic network node. To support a transportable solution, we use a small compact aluminium case which hosts a small motherboard (200x150 mm) featuring a VIA C3 CPU 1GHz (supporting the x86 instruction set), 256MB DDR RAM, 3 Giga Ethernet LAN port, 2 PCMCIA slot, 4 USB port and one serial port.

To reduce risk of failure, we choose a fan less hardware solution. Moreover, the box does not embed a mechanical hard disk drive. The operating system, file system and execution environment are stored in a memory card (e.g Compact Flash).

Figure 1 shows, on the left, an inside view of the case where we can see the small mother board and its large passive cooling system (white part) hiding chipset and CPU. A second picture shows a backside view of the case with all connectors.

Fig. 1. Internal and connections views of the industrial autonomic network node.

3 Software execution environment

3.1 Operating System

The industrial autonomic network node environment runs over *Btux* provided by the Bearstech[2] company. Btux is based on a GNU / Linux operating system running a 2.6.12 kernel version. However the whole system has been rebuilt from scratch and designed for embedding systems (small memory footprint, selected command set available). The operating system respects standards and is remotely upgradeable to easily apply patches and updates without human intervention.

For the IAN^2 node, we worked in tight collaboration with the Bearstech engineers to add a secured network wireless connection and multimedia sensors (audio and video) support. We also use information returned by internal sensors (i.e temperature) in order to take intelligent decision and do not expose our hardware to critical situation (e.g over heating).

3.2 Programmable dynamic environment

This section describes the software used on top of the operating system node (described above). This software is called an Execution Environment (EE) which is used to dynamically plug-in and run Active Applications (AA) also called active services. A service is deployed on demand and applied on one or several data streams. Services can run in parallel and are all executed in the EE.

***IAN*^{2} software architecture** We propose the IAN^2 Industrial Autonomic Network Node architecture (Fig. 2). This node supports switching and routing protocols through wire and wireless connection hardware. The limited CPU facilities are open to dynamically deploy autonomic services. Some limited storage capabilities are available to support heterogeneous classes of services.

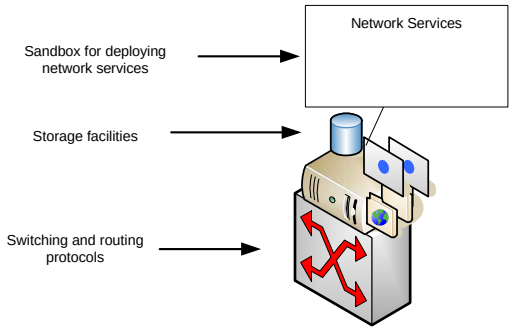

Fig. 2. IAN2 : Industrial Autonomic Network Node

Our Execution Environment called $Tamanoir^{embedded}$ is based on the Tamanoir [7, 8] software suite written by L. Lefèvre and J.P. Gelas (from INRIA, France). The Tamanoir suite is a high-performance execution environment for active networks designed to be deployed in either local area networks or wide area networks. It is a prototype software with features too complex for an industrial purpose (cluster-based approach, Linux modules, multi-level services[11]...).

Due to some typical industrial constraints (e.g code maintenance), we reduce the code complexity and remove all unused classes and methods or actually useless for the Temic [3] project. It allows us to reduce the overall size of the software suite and make the maintenance and improvement of the code easier for service developers (Figure 3).

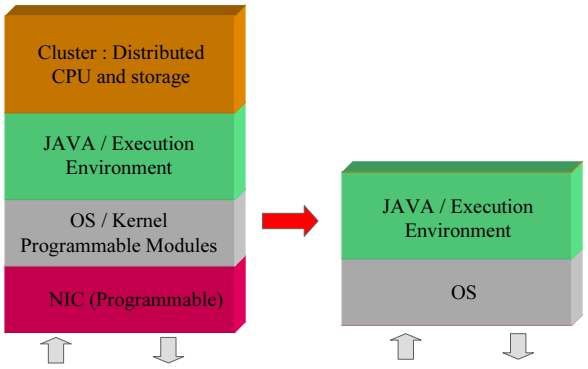

Fig. 3. From a generic Active Network environment (Tamanoir) to an Industrial Autonomic Environment ($Tamanoir^{embedded}$)

$Tamanoir^{embedded}$ is a dedicated software platform fully written in Java and suitable for heterogeneous services. Tamanoir provides various methods for dynamic service deployment. First method allows services to be downloaded from a service repository to a Tamanoir Active Node (TAN). Second method allows a TAN to request the service from the previous active node crossed by the active data stream (Figure 4).

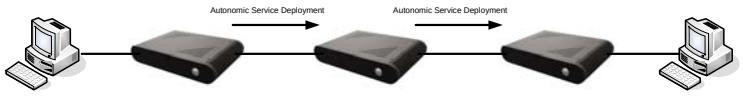

Fig. 4. Autonomic Service Deployment on wire connections

$Tamanoir^{embedded}$ also supports autonomic deployment and services updating through mobile equipments (Figure 5). Inside automatic maintenance

projects, we deploy wireless based IAN^2 nodes in remote industrial environments (no wire connections available). In order to download maintenance information, human agents can come near IAN^2 nodes to request informations. During this step, mobile equipments (PDA, Tablets, cellulars) are also used as mobile repositories to push new services and software inside autonomic nodes (Figure 5).

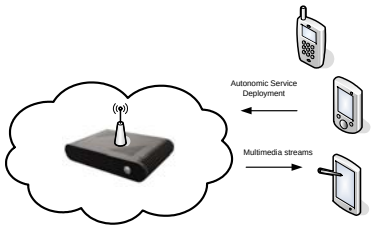

Fig. 5. Autonomic Service Deployment through mobile nodes

4 Experimental evaluation

In this section, we present first evaluations of the industrial autonomic network node. Experimental results are divided into three categories. First, we present results in terms of network performances (wire and wireless networks). Then, we explore some preliminary results obtained with the software of the autonomic execution environment of the IAN^2. And last, we present experimental results obtained in a multimedia industrial context.

4.1 Network performances

We evaluate the performances of IAN^2 concerning wire and wireless network interfaces. We used the `iperf`[1] tool for measuring TCP bandwidth performances. Iperf can report bandwidth, delay jitter and datagram loss. We experiment network performances within two topologies (Figure 6).

Fig. 6. *back-2-back* and *gateway* experimental local platforms

We call *back-2-back* topology when one IAN^2 is connected straight to another IAN^2 through a short (50 cm) Ethernet cable (cat 6) (Figure 6). We call

gateway topology when we connect two IAN^2 through a third one. In this case we allow IP forwarding on the node in the middle. We set the TCP no delay IPerf option which disables Nagle algorithm but we didn't noticed any significant difference. Table 1 shows bandwidth results and corresponding CPU usage under the two different topologies. We observe that *back-2-back* IAN^2 nodes failed to obtain a full Gbit bandwidth with TCP streams. When a third node is involved as a *gateway*, throughput is also more impacted. These results come mainly from the limited CPU embedded in the IAN^2 which limits the capabilities of sending large streams of data.

Configuration.	Throughput	cpu send	cpu recv	cpu gateway
back-2-back	488 Mbps	90%	95%	N / A
gateway (1 stream)	195 Mbps	29%	28%	50%
gateway (8 streams)	278 Mbps	99%	65%	70%

Table 1. Raw performances shown by *iperf* with default values of buffer length and TCP window size.

For the next experience, we use one industrial autonomic network node ($IAN^2$1) to transmit two streams to another industrial autonomic network node ($IAN^2$2), using only one Giga Ethernet link. We obtain 312 Mbps and 229 Mbps (total 541 Mbps) and each CPU was used to the maximum (2x50% on transmitter and receiver). We also try the full-duplex feature of our card by sending one stream from $IAN^2$1 to $IAN^2$2 and *vice-et-versa* (bidirectional connection). We obtain 196 Mbps and 247 Mbps (total 443 Mbps). About 2x50 % of CPU was used on each CPU (i.e transmitter and receiver). We notice that there are as many iperf processes running as data streams on the link, and each process shares with equity the CPU load. The figure 7 shows throughput performance reduction in function of number of streams on a back-to-back connection topology.

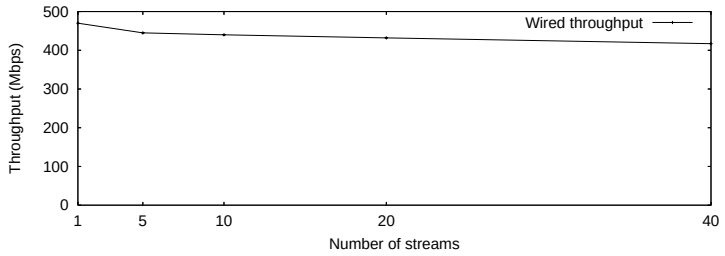

Fig. 7. Throughput performance reduction when number of stream increases between two IAN^2 connected back-to-back.

We also did some tests with a PCMCIA wireless card (using the Orinoco Linux modules) plugged in the IAN^2 (802.11b). Best obtained throughput, when the IAN^2 is 10 meters far from the wireless Access Point was only 4.45 Mbps (without external antenna).

The figure 8 shows throughput performance reduction in function of number of streams in a wireless context. We also did some bidirectional tests and surprisingly we obtain an average throughput equal to the maximum speed. Finally, we try to remove the TCP no delay option (Nagle) and obtain a slightly lower performance (3.92 Mbps).

These experiments show that IAN^2 nodes must be dedicated to some specific platforms (wireless environments, xDSL, Fast Ethernet). This can be compatible with some current industrial deployments.

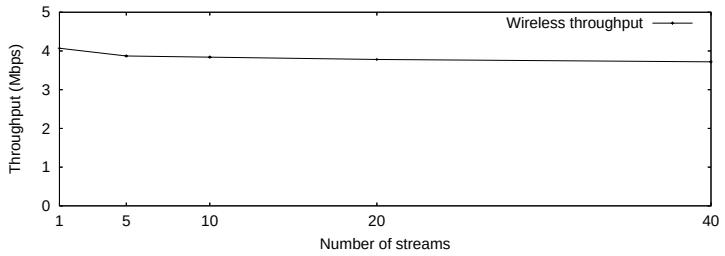

Fig. 8. Throughput performance reduction when number of stream increases in a wireless context.

4.2 Evaluating autonomic performances

We present some results obtained with the $Tamanoir^{embedded}$ Execution Environment. We ran two different active services : a lightweight service (in terms of CPU usage), called *MarkS*, used to count and mark the packets crossing the Tamanoir node. And a heavyweight service (in terms of CPU usage), called *GzipS* which compresses packets payload on-the-fly using the Lempel-Ziv coding (LZ77). The Execution Environment and services run in a SUN JVM 1.4.2. Table 2 shows performance results with different payload size for both services.

	4kB	16kB	32kB	56kB
MarkS	96	144	112	80
GzipS	9.8	14.5	15.9	16.6

Table 2. Throughput (Mbps) of Tamanoir applying a lightweight service (MarkS) and a CPU consuming service (GzipS).

We compare obtained results with high performance active network node platform (embedded in a Compaq DL360 (G2) Proliant, dual-PIII, 1.4GHz, 66MHz PCI bus). We used different network interfaces (Fast, Giga Ethernet) and protocols (UDP / TCP).

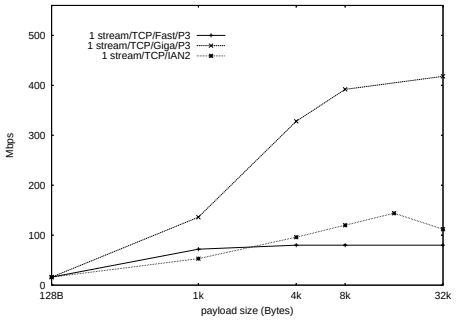

Fig. 9. Throughput comparisons of a lightweight service

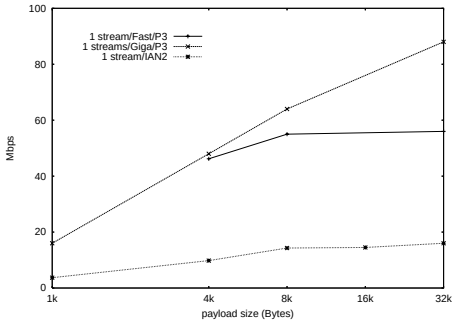

Fig. 10. Throughput comparisons of a heavy service

The IAN^2 based on a reliable fan-less no disk node with a lightweight Execution Environment, shows comparable results with a slow desktop machine with a slow hard disk drive. We can see the limit of IAN^2 CPU with the *GzipS* service (CPU is 100% used for a bandwidth of 16 Mbps). For a lightweight service, like the *MarkS* service, we observe a combined limitation of the CPU and the IAN2 network interfaces cards. The major point of deception is about the network interfaces announced to support 1 Gbps and which sustain with difficulty only half of the bandwidth. Moreover, we can observe that due the modification of the Tamanoir high performance Execution Environment, the autonomic node does not benefit from some improvements (lightweight Linux modules, efficient JVM...). Thus all data packets are processed inside the Java Virtual Machine. But the industrial deployment on an industrial autonomic network node can still benefit from this trade off between performances and reliability.

4.3 Performances of the IAN^2 node within multimedia context application

In our architecture, the industrial autonomic node is the point where all active services are performed, so it is a critical point. To evaluate its performances, we measure the processor load during the adapting and the transmitting of a video file (Fig 3).

Results show that the CPU of the IAN^2 (VIA C3 1 GHz) is intensively used during video adapting. There is no enhancement even when the video size decreases. In table 3, the transmission with the MJPEG format is performed by the

Format / Size	Usr CPU load
MJPEG / 720x480	< 1 %
H263 / 352x288	98,7 %
H263 / 176x144	99,3 %
H263 / 128x96	99 %

Table 3. CPU load on the IAN^2 when adapting and transmitting a video file

same active service, but with no adapting step. In this case, we observe that the CPU load is negligible. This proves that the load is totally due to the processing of data adapting and resizing, done by the autonomic service. Adapting is computation intensive for the IAN^2 equipment..

To evaluate the impacts of data adaptation on the network, we measure the output data rate on the active node (using wireless network), when transmitting an adapted video file to a PDA (table 4) and when transmitting over RTP an adapted video stream to a laptop (figure 11).

Output Format / Resolution	Entry File / Output File	Transmitting time	PDA loading time
MJPEG / 720x480	14794 KB / 14794 KB	4 min 50 sec	5 min 10 sec
H263 / 352x288	14794 KB / 1448 KB	22 sec	2 min 55 sec
H263 / 176x144	14794 KB / 365 KB	8,5 sec	1 min 30 sec
H263 / 128x96	14794 KB / 179 KB	3,8 sec	1 min 18 sec

Table 4. Output data rate when adapting and transmitting a video file on IAN^2

We can observe on table 4 the link between video file adaptation (requiring an intensive usage of CPU) with transmitting performances of adapted file. The IAN^2 node applies adaptation of a MJPEG file and sends the results to a remote PDA. Even with a limited CPU, the industrial autonomic network node provides efficient adaptation which reduces the amount of transported data and globally improves the performances of the application.

On figure 11, we also observe the same results : the adapted stream consumes less bandwidth when changing the video format (from MJPEG to H263) and when resizing video (from CIF 352x288 to SQCIF 128x96). Resizing the video decreases considerably the amount of data transmitted on the network. When transmitting a multimedia file, the transmitting time is thus shortened, and so the occupation time on the network. These results show that adaptation on the autonomic network node is beneficial to save network bandwidth.

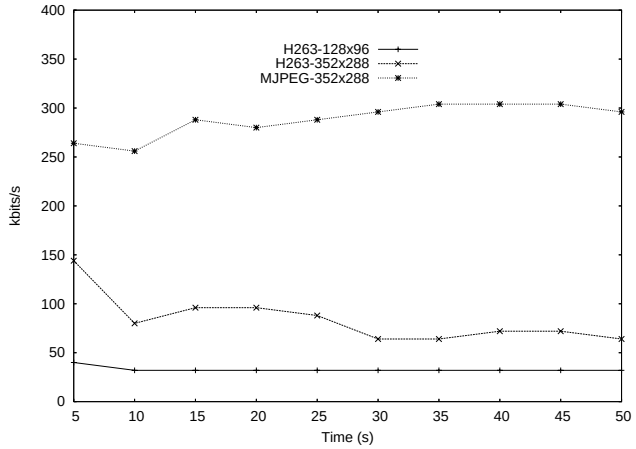

Fig. 11. Output data rate when adapting and transmitting a video stream over RTP.

5 Related works

Various research projects linking academic partners and industrial partners have explored the design of industry targeted active network and environments. GCAP[15] describes a prototype of an active commercial router. While ANDROID[4] and FAIN[5, 6] propose models and prototypes of generic active network architecture.

Some companies have also explored the proposition of active infrastructure in order to deal with their own needs. NTT[13] proposes the A-BOX an active network node mixing hardware and software for supporting Gigabit networks. Hitachi[6, 14] presents some gateways mixing active networks and web services. Few network operators have experimented deployment of active environment on their network equipments. Nortel has proposed a software execution environment running on an Accelar Gigabit switch[9, 10]. Alcatel[12] has proposed prototype of an active node based on an OmniSwitch Alcatel router.

6 Conclusion and Future works

In this paper we described the design of the IAN^2 prototype of industrial autonomic network node. We discuss hardware choice to fit typical industrial requirements in terms of space usage, limited accessibility and then remote maintenance. Then, we discussed the software solution used to provide a reliable network device based on a minimal open source operating system easily upgradeable remotely. We proposed the $Tamanoir^{embedded}$ suite, a simplified execution environment able to dynamically deploy active services and apply them to the data streams in order to adapt them to the terminal clients (e.g tablet PC, cell phone or PDA). By doing this, we propose a reliable “plug and route” autonomic node.

This paper provides also a performance evaluation of IAN^2 in terms of processing power, networking and Execution Environment performances. Results show that performances are far from a current desktop machine so we cannot deploy IAN^2 for high performance networking (Giga) platforms. But for lower bandwidth architecture (Fast Ethernet, xDSL or Wireless networks), IAN^2 nodes can perfectly support a large class of reliable autonomic services.

Switching from an experimental and academic project to an industrial project providing an equipment running in a production context is a real challenge. However, it is a mandatory step if we want to see one day a large number of active and programmable equipments deployed.

Next step concerns the development of a set of autonomic services in order to be deployed on IAN^2 nodes.

Acknowledgments

The authors like to thank the members of the RNRT TEMIC project : SWI company, Université de Franche Comté (LIFC) and the Université de Haute-Alsace (GRTC). The authors like also to thank L. Haond and L. Montagne from Bearstech[2] company who provide valuable help with Btux.

References

1. Iperf. http://dast.nlanr.net/Projects/Iperf/.
2. Bearstech company. http://bearstech.com, 2005.
3. Tobiet et al. Panorama des réseaux utilisés et services à valeur ajoutée temic. Delivrable D2.1 - http://temic.free.fr, March 2005.
4. Mike Fisher. Android : Active network distributed open infrastructure development. Technical report, University College of London, 2001.
5. Alex Galis, Spyros Denazis, Celestin Brou, and Cornel Klein. *Programmable Networks for IP Service Deployment*. Artech House, May 2004.
6. Alex Galis, Bernhard Plattner, Jonathan M. Smith, Spyros G. Denazis, Eckhard Moeller, Hui Guo, Cornel Klein, Joan Serrat, Jan Laarhuis, George T. Karetsos, and Chris Todd. A flexible IP active networks architecture. In *International Working Conference on Active Networks (IWAN 2000)*, pages 1–15, 2000.
7. Jean-Patrick Gelas, Saad El Hadri, and Laurent 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.
8. Jean-Patrick Gelas, Saad El Hadri, and Laurent Lefèvre. Towards the design of an high performance active node. *Parallel Processing Letters journal*, 13(2), June 2003.
9. R. Jaeger, S. Bhattacharjee, J. Hollingsworth, R. Duncan, T. Lavian, and F. Travostino. Integrated active networking and commercial-grade routing platforms. In *Usenix: Intelligence at the Network Edge*, San Francisco, March 2000.
10. Tal Lavian and Phil Wang. Active networking on a programmable networking platform. In *IEEE OpenArch'01*, Anchorage, Alaska, April 2001.

11. Laurent Lefèvre. Heavy and lightweight dynamic network services : challenges and experiments for designing intelligent solutions in evolvable next generation networks. In IEEE Society, editor, *Workshop on Autonomic Communication for Evolvable Next Generation Networks - The 7th International Symposium on Autonomous Decentralized Systems*, pages 738–743, Chengdu, Jiuzhaigou, China, April 2005.
12. Olivier Marcé, Laurent Clevy, Carlo Drago, and Olivier Le Moigne. Toward an industrial active IP network. In *Third International Working Conference on Active Networks (IWAN)*, Philadelphia, USA, September 2001.
13. Takahiro Murooka, Masashi Hashimoto, Noriyuki Takahashi, and Toshiaki Miyazaki. High-speed active network node - its concept and implementation for gigabit-network applications. In *The 47th IEEE International Midwest Symposium on Circuits and Systems*, 2004.
14. Takashi Nishikado, Minoru Koizumi, and Hideo Oochi. Large-scale high-quality communication service solution using active network technology. In *Hitachi Review*, volume 49, December 2000.
15. M. Urueña, D. Larrabeiti, M. Calderón, A. Azcorra, J. E. Kristensen, L. K. Kristensen, E. Exposito, D. Garduno, and M. Diaz. An active network approach to support multimedia relays. In *Joint International Workshop on Interactive Distributed Multimedia Systems / Protocols for Multimedia Systems (IDMS-PROMS)*, Coimbra, Portugal, November 2002.