Adaptation of control parameters based on QoS monitoring

Francis Lepage, Thierry Divoux, Fabien Michaut

Adaptation of control parameters based on QoS monitoring

F. LEPAGE, T. DIVOUX, F. MICHAUT
CRAN, *CNRS UMR 7039, Université Henri Poincaré NANCY I BP 239*
54506 Vandoeuvre-les-Nancy Cedex, FRANCE. E-mail: francis.lepage@cran.uhp-nancy.fr

Abstract : The teleoperation of robotic systems through heterogeneous networks such as the Internet are more and more current nowadays. The various delays of data transfer with this kind of networks have to be taken into account in the control/observation loop. This paper presents a Quality of Service (QoS) architecture dedicated to control parameters adaptation to resources fluctuations and its use for the teleoperation of a mobile robot. The experimental results show the validity of the approach.

Keywords : Adaptation, Quality of Service, teleoperation

I. INTRODUCTION

Various new networking applications have appeared in the past ten years. Networks initially designed for the transfer of data without time constraints are used nowadays for more complex data traffics as audio, video or telecontrol with strong constraints in terms of delay, jitter, bandwidth, etc. Therefore, the Quality of Service (QoS) of communication systems and more generally the end-to-end QoS (including networks and end-systems) need to be managed in order to make these new applications operating properly. In this context, many authors have proposed architectures to deal with QoS. Most of these architectures assume that resources guarantees can be provided to applications and then consist in reserving resources according to applications needs. It is the case when applications are distributed on Local Area Networks for which one checks a priori the sufficiency of resource and the respect of QoS parameters. But resources reservation is in fact very difficult to implement in the global Internet and impossible to make for a specific data flow. It is why applications have to adapt themselves to the operating environments [1]. In this approach we have designed an architecture adapting the application to the actual QoS. It is based on the 'Prayer' architecture introduced by Barghavan and Gupta in [2]. It enables to changes the control parameters in order to have the better control loop conditions.
This paper is organized in three sections: the first introduces adaptation mechanisms in distributed systems, and the second presents the proposed adaptation architecture. The last section is dedicated to its use for adapting control parameters in an application where a mobile robot is remotely controlled, and gives some experimental results.

II. ADAPTATION IN DISTRIBUTED SYSTEMS

A. Adaptative control

The adaptive control is a whole of techniques used for real time tuning of the regulators of the control loops in order to carry out or maintain a certain level of performances when the parameters of the process are unknown and/or vary. Its principle is schematized on figure 1.

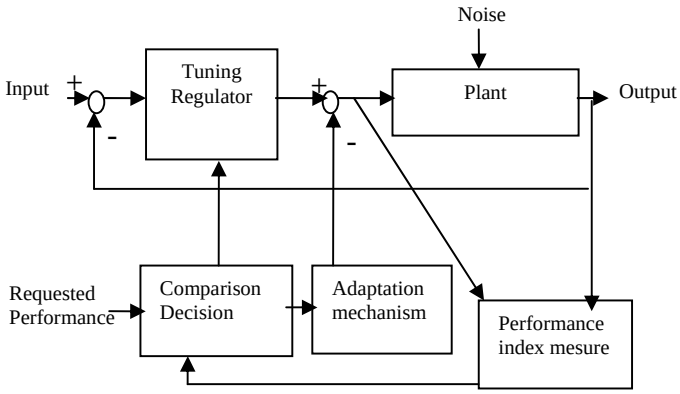

Fig. 1. Adaptative control principle

The direct use of adaptive control in distributed systems is not easy according the transmission delays. For example the measurement of the index of performance is received after a variable delay. It could even be difficult to make coherent, i.e. to be dated from the same moment, if the process is itself distributed. This technique was not chosen because of its too specific constraints but could be an interesting control mechanism using synchronization of the scattered elements by using GPS for example.

B Different adaptation levels

The adaptation to the current operating parameters can be made at three levels of the distributed system architecture: the system level (equipment system and communication system), the middleware level (intermediate level) and the application level.
The system level is illustrated perfectly by the flow control of TCP (Transport Protocol Control) used in the Internet, rather known in various communities [3]. To control the traffic in the network nodes crossed by a communication, the transmitter adjusts its flow according to the Round Trip Time and of the loss of acknowledgment packets. The general idea is thus close to the adaptive control principle, one tunes a regulator according to the measured performances. It is a very efficient technique if all (or a very large majority) of the flows transmitters use the same type of control. If not, the transmitter penalizes its flow with the profit of the others without improving the regulation of the traffic in the crossed nodes.
The other adaptations made to the system level concern: the choice or the adaptation of the protocol like in the example given in [4], the scheduling of the packets, the modification of the size of the packets and finally the choice of the network. The network management and the scheduling in operating systems implement in fact these adaptations.

The adaptation at the middleware level consists in implementing either filtering or storage between the application and the system [5].

The adaptation to the Application level consists in adapting the needs for the application to QoS offered by the network. This adaptation can concern the nature of information: if for example the throuput becomes insufficient to transmit images, a textual description of them is transmitted. It can relate to the modulation of the throuput with a change of codec or compression ratio [6]. Lastly, it can relate to the adjustment of the buffer size used to synchronize the samples of information of continuous nature i.e. time dependent (voice, video) [7].

C. Technique of adaptation blocks

The technique of adaptation blocks operates at the Application level [2]. It initially consists in directly taking into account a measure of the operating environment of the application rather than its consequences. For a system distributed on a network, it is a question of measuring end to end QoS since its attributes are the parameters of the environment of the application.

Moreover, the adaptation mechanism is based on the concept of QoS classes and execution sequences. A QoS class is a domain of QoS parameters values bounded by minimal and maximal values. The execution sequence principle consists in dividing the control execution in intervals. A sequence is an execution with a set of parameters values. Notice that due to the finite and limited number of sequences the number of parameters values is also limited.
The different control application execution is divided into adaptation blocks. Each block is as set of alternative execution sequences and is associated with a QoS class (see figure 2).

Starting an adaptation block, the application negotiates a QoS class and executes the corresponding execution sequence. This process is modelled by a Petri net. The Petri net in Figure 3 is a case of three execution sequences for the next block.

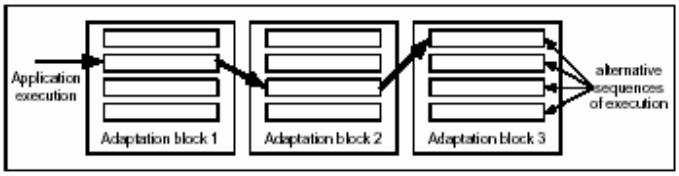

Figure 2. Adaptation blocks.

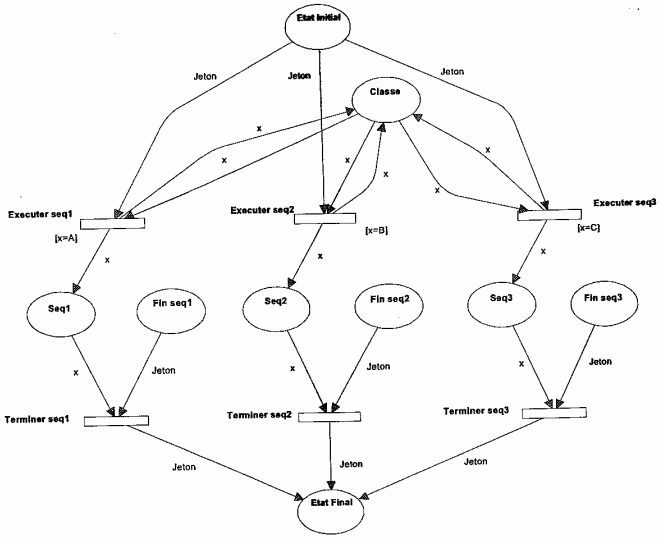

Figure 3. Petri Net model of sequences choice

In order to be more flexible and to follow quick QoS level changes, an additional adaptation mechanism is used. It acts during a execution sequence running especially when the QoS level decreases. The resource management layer tries to provide QoS bounds during the adaptation block execution using resources reservation mechanisms. If the QoS class is

violated or if the resource management can provide a better QoS class, the resource management layer notifies the application.

The application consequently reacts according to 'fallback' actions which are associated with the current execution sequence. Fallback actions are of two types: DOWN actions or UP actions. DOWN actions define fallback actions to initiate when the QoS class is violated. UP actions specify fallback actions if a higher QoS class can be satisfied, but are seldom used.

Examples of DOWN fallback actions are BLOCK (causes the application to stop its execution up to the QoS class can be provided again), ROLLBACK (re-executes the current adaptation block), ABORT (aborts the current adaptation block). Examples of UP fallback actions are ROLLBACK (with the new QoS class) and SWITCH (go to another sequence).
This adaptation method enables applications to provide the adaptation rules. But the adaptation mechanism is provided by the architecture according to the Campbell transparency principle [8].

III. QoS - ADAPTATION ARCHITECTURE

The proposed new architecture is described in figure 4. It is called QoS-Adapt. It is an improvement of the Prayer architecture by adding crucial elements permitting its real working:

- a QoS monitoring agent with a metrology service [9]
- QoS class concept is extended to the different layers of the architecture
- A user interface permits to enter user preferences
- mechanisms are added to control the emitted flow.

It is organized in three layers like in the Prayer architecture: resources layer, resource management layer, and application layer.
Resources include networking resources (usual protocol stack: transport layer, network layer, etc.), and systems resources (CPU, memory, etc.). According to its name, the management resource layer manages these resources and provides the application layer with adaptation mechanisms.

At the application layer, a module is associated to each application. A module consists in four components: the user interface component, the adaptation component, the QoS classes component and the flow policing component. The user interface provides the user with different configurations in which the application can operate. The adaptation component is the interface which the resource management layer uses to interact with the application execution. The QoS classes component supplies the list of all the QoS classes provided by the application. At the end, the flow policing component is used to ensure that the application generates a traffic that complies with its expected specifications.
The resource management layer consists in components and modules. The three components are the manager, the scheduler and the QoS mapping component. The manager controls execution adaptations of all applications that are running on the end-system. It interacts directly with the adaptation component of the application and enables competing applications to use available resources according to the scheduler component. The scheduler defines priorities between applications according to user preferences (the user specifies previously the application priority level in the application user interface component). The QoS has to be considered and specified at each layer of the architecture. Therefore a translator is needed. This translator is the QoS Mapping component.

Each module corresponds to an application. A module consists in three components: the QoS monitoring component that performs QoS measurements, the notification component that informs the manager component about the measured QoS and the traffic control component. The latter is carrying out measurements on traffic generated by the application and works with the flow policing component (at the application layer) to ensure that the application behaves properly. Note that our work is not interested in resources reserving but only in adapting application execution.

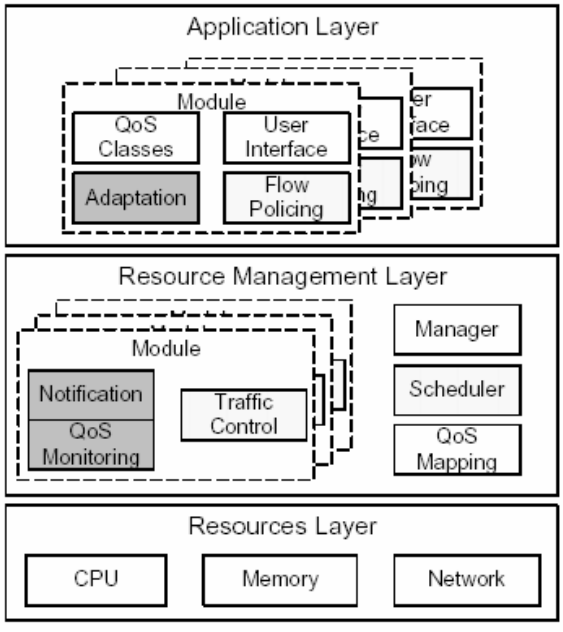

Figure 4. QoS Adapt architecture

IV. APPLICATION TO THE REMOTE CONTROL OF A ROBOT

The platform described in the figure 5 is composed of a mobile robot, of a camera, and of an embedded PC. The embedded PC dialogs with the robot through a serial line. The embedded PC is interconnected to the wired network via a 802.11b wireless channel. A camera sends an analogical video signal by using a radio-frequency transmitter to an http server which digitalises this signal and displays JPEG images.

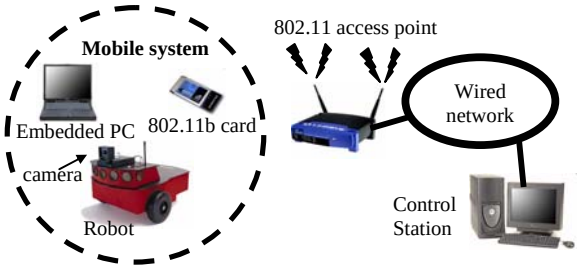

Figure 5. The experimental platform

The goal is to control the robot position x from the control station. The robot is speed controlled. It measures its position by using sensors (odometers) and sends it back to the control station which computes a speed feedback input s. This latter is transmitted to the robot. A proportional corrector is implemented on the control station. The application cannot directly dialog with the robot sensors and actuators, but with the controller card which is a closed system for the user.

The QoS-Adapt architecture and the control application are developed in C++ over Linux. The application is shared between the embedded PC and the control station (fig 6):

- on the control station, it receives the measured position x of the robot, computes the feedback input s, and sends it to the embedded PC;
- on the embedded PC, it requests the robot for its position x, and forwards this information to the control station. It also receives the feedback input s from the control station and forwards this information to the robot.

Using the network for the transmission of both the control data and the measures of the robot position introduces delays in the control loop. For this kind of applications, disturbances introduced by the network have been studied: [10] defines a delay regulator in order to deal with constant delays for which the system stability is easier to obtain. [11] computes a performance criterion (phase margin), regarding the delays and the losses due to the network use. The impact of the delay on our control is detailed now.

In the following, forward and backward delays are assumed to be equal to the half of the Round Trip Time (RTT).

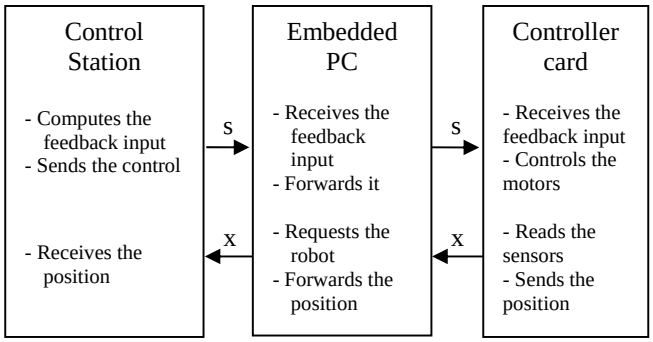

Figure 6. Control distribution

This system is simulated with Matlab. The robot has been previously identified (figure 7) by using the CONTSID Toolbox [12], and its transfer function is :

$$F(s) = \frac{16{,}6117.k.e^{-\left(70.10^{-3}\right)s}}{s^3 + 4{,}0450s^2 + 16{,}5611s}$$

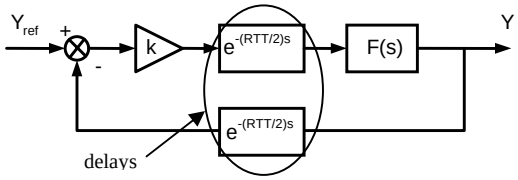

Figure 7. The robot control model

The response to a step is simulated for different values of the RTT. Delays induce greater and greater overshoots. This can be dangerous when the robot moves near obstacles because collisions could occur.

It is well known that a solution to suppress the overshoots is to decrease the value of the gain k. One way is to compute the gain for the maximum delay. But this approach makes the system too slow. For example (Table 1), to suppress the overshoots for delays less than 1s, the response time t_r grows from 1.38s (k=0.9) up to 6.53s (k=0.3). So, when the delay is less than its maximum value, the system is needlessly slow.

k	0.3	0.5	0.7	0.9
t_r (s)	6.53	3.59	2.37	1.38

Table 1

That is why it is interesting to dynamically adapt the gain regarding predefined slots of the delay.

The application is developed in conformance with the QoS-Adapt architecture. For that, four QoS classes have been defined. Each class corresponds to an applicative situation, ie to a value of k, regarding the measured value of the delay. Table 2 presents the maximum delays computed to avoid overshoots for different values of k corresponding to the four QoS classes. One adaptation bloc is periodically repeated. It is constituted of four execution sequences in which the control is computed during the bloc period for each QoS class.

QoS Class	A	B	C	D
Gain (k)	0,9	0,7	0,5	0,3
RTT max (s)	0,1	0,3	0,5	1,1

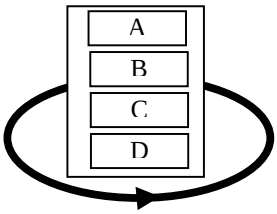

Table 2

Remark 1: the delay values are greater than those we can observe nowadays on the internet. This is due to the extreme slowness of the studied system. But that does not re-opens our proposals: a better robot (moving at several m/s) would lead to lower delays in conformance with those measured on the internet.

Remark 2: the QoS-Adapt architecture includes a QoS monitoring component described in [9]. Up to now, it only implements active measurement techniques. These techniques generate on the network an additional traffic, in opposition with passive techniques which are only based on the analysis of the application traffic. For this experimentation, we have not used this monitoring component because passive measurements seem to be more appropriated: here the delay is determined regarding the dates when the packets carrying the feedback input and the position information. Those which are sent by the embedded PC are time-stamped at the beginning of their transmission. When the control station receives them, it computes the feedback input and sends it to the embedded PC with the previous time-stamp. This latter deduces the RTT by subtraction of the packet reception time and of the received time-stamp.

Three adaptation actions are defined (table 3). The UP actions are launched when a better QoS class can be satisfied. Note that in this specific case, no real UP action is made. The DOWN actions are launched when the current QoS class is no more satisfied. Three actions are described in this paper:

QoS class	UP action	DOWN action
A	-	SWITCH
B	BEST_EFFORT	SWITCH
C	BEST_EFFORT	SWITCH
D	BEST_EFFORT	BLOCK

Table 3. Adaptation actions

-BEST_EFFORT : the control is computed without any modification of the gain during the whole bloc period. This action avoids oscillations between to sequences when RTT evolves around the limit of 2 QoS class.
- SWITCH (from a sequence to an other) : the value of k is modified during the bloc period
-BLOCK : the application is stopped until the QoS class becomes available again.

Remark: no UP action is defined for the sequence which corresponds to the QoS class A because no higher class is available.

The figures 8 and 9 describe the parallel evolution of the RTT, and of the gain. The figure 10 shows how the system follows the computed control. To note that the robot stops 2 times (k = 0), when the RTT becomes greater that 1,10 s.

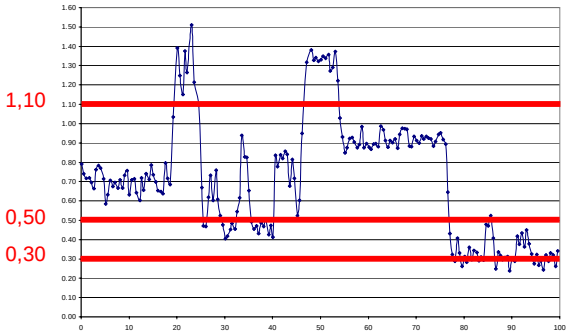

Figure 8. RTT evolution

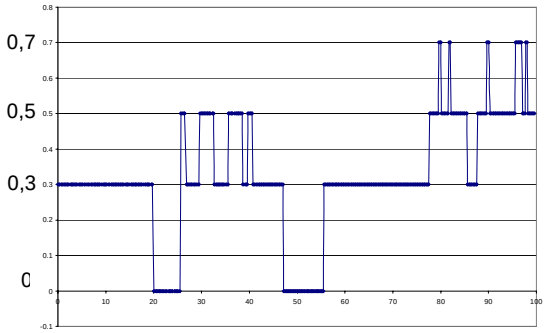

Figure 9. Gain evolution

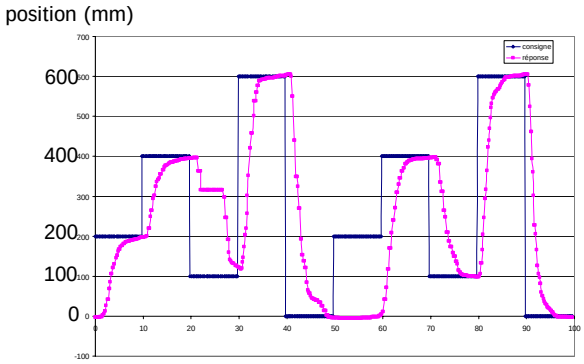

Figure 10. Response to the control

V. CONCLUSION

Teleoperation through a packet switching network whose quality of service is not mastered requires the use of adaptation mechanisms of the control. This paper briefly describes an architecture framework dedicated to the applications which are sensitive to QoS fluctuation. This architecture is detailed in [13] in which the adaptation method has been formally specified.

Experimentations have shown that this architecture makes possible the implementation of adaptive control strategies in order to react to the delays induced by the network, and so to avoid the overshoots of the system.

The adaptation strategy presented in this paper is not necessarily the best in all cases. It has been chosen because of its simplicity and of its universality. The possibilities provided by the architecture could enable the use of more efficient strategies for specific applications, such as the remote position control described in this paper.

VI. REFERENCES

[1] C. Diot, C. Huitema, and T. Turletti, “Multimedia Applications should be Adaptive,” Proc. HPCS’95, Mystic, CN, 1995.

[2] V. Bharghavan and V. Gupta, “A Framework for Application Adaptation in Mobile Computing Environments,” Proc. IEEE Compsac’97, Novembre 1997.

[3] F. Baccelli, D. Hong, "TCP is Max-Plus Linear," SIGCOMM 2000, Stockholm, Sweden, 2000.

[4] A. Friday, N. Davies, GS. Blair, et KWJ. Cheverst, "Developing adaptative applications : the MOST experience. Integrated Computer-Aided Engineering, 6(2) : 143-157, 1999.

[5] A. Schill, S. Kümmel, T. Springer, et T. Ziegert, "Two approaches for an adaptative multimedia transfer service for mobile environments," Computer Graphics, 23 :849-856, 1999.

[6] M. Fry et A. Ghosh, "Application level active networking", Computer networks, 31 : 655-667, 1999.

[7] K. Fujimoto, S. Ata and M. Murata, "Adaptative playout buffer algorithm for enhancing perceived quality of streaming applications", IEEE GLOBECOM, 2002.

[8] A. T. Campbell, A Quality of Service Architecture, PhD thesis, Computing Department, Lancaster University, 1996.

[9] F. Michaut, F. Lepage, “A tool to monitor networks QoS,” IEEE/IFIP Conference on Network Computer and Engineering. NETCON’02, Paris, Octobre 2002.

[10] A. Lelevé, “Contribution à la téléopération de robots en présence de délais de transmission variables,” Thèse de l'Université de Montpellier II, 2000.

[11]. G. Juanole, “Quality of Service of communication networks and distributed automation : models and performances“, 15th IFAC World Congress, Barcelona, July 2002

[12] H. Garnier, M. Mensler, “The CONTSID toolbox: a Matlab toolbox for CONtinuous-Time System Identification,” 12th IFAC Symposium on System Identification, Santa Barbara, USA, Juin 2000.

[13] F. Michaut, "Adaptation des applications distribuées à la Qualité de Service fournie par le réseau de communication", thèse de doctorat de l'Université Henri Poincaré, Nancy1, 26 novembre 2003.