Teleoperation through the Internet : Experimental Results with a Complex Manipulator

Arnaud Lelevé, Philippe Fraisse, Pierre Dauchez, François Pierrot

Teleoperation through the Internet : Experimental Results with a Complex Manipulator

Arnaud LELEVE, Philippe FRAISSE, Pierre DAUCHEZ, François PIERROT

LIRMM - UMR 5506 CNRS / Université Montpellier II
161 rue Ada - 34392 Montpellier CEDEX 5 - France

Abstract: According to their research activities, some laboratories have to set up and share expensive full-scale experiments requiring the use of teleoperated robots. It is in this context that we have built a teleoperation structure designed to study the problems usually encountered in such experimentations, We have first built a generic simulation model that guided us in the realization of a real teleoperation platform using a terrestrial mobile manipulator. This paper summarizes the results of the simulations we performed and presents the first results we obtained with the real platform.

1. PRELIMINARIES AND PROBLEM DESCRIPTION

1.1 Experiment Sharing

In a context of sharing of experiment platforms, we have been led to build a teleoperation through the *Internet* platform. This kind of project could be interesting for every laboratory liable to make experiments outside its walls. For example, underwater data collecting in the ocean driven from one own's lab [1] or some work in spatial environments [2]. As such an experimentation is heavy and time and money expensive, the use of a common medium such as the *Internet* would allow several laboratories to share the collected data and to collect their own data without having to reorganize the whole experiment.

1.2 Technical Problems

The long distance control of a remote system requires the use of different means of information media which causes two main technical problems in teleoperation : limited bandwidth and sometimes high communication delays due to the propagation, packetisation and the many other events the *Internet* relays may inflict on the data [3]. Moreover the bandwidth and the delays may vary according to the events that occur all along the transmission lines. In acoustic transmission, round-trip delays greater than 10s and rates smaller than 10kbits/s are common.

1.3 Logistics Problems

These technical restraints result in difficulties for the operator to securely control the remote system. If he has to wait at least 10 seconds before being able to see the result of his action, it is very difficult for him to perform his task and in case of trouble, he may notice it too late. It seems necessary to add a minimum of autonomy in the teleoperated system to avoid critical situations.

Moreover, the bandwidth can prevent the use of high-resolution real-time video views of the experimentation site. This requires the man-machine interface to display a 2D/3D model of the experimentation site, which cannot be accurate when the operating site is ill-known or evolves.

1.4 Common Long Distance Teleoperation Techniques

When the bandwidth is small and the delays are prohibitive, the common solution is to send macro-instructions to the remote system and wait for its completion. Recent experiments showed remote robots able to follow a pre-loaded (possibly dynamic) trajectory, grip a cubic piece with force control, … [4]. Briefly speaking, the robot becomes momentarily autonomous with respect to the goal commands it continuously receives.

Efforts in improving the man-machine interface [5] have provided the ability for the operator to previously simulate the macro-instructions before sending them and to match a virtual reality model (regularly refreshed by the feedback data) of the performing robot with a video stream [6] (the available bandwidth may limit the stream to periodic pictures of the experiment site). Virtual Reality instruments as 3D helmets and glove sensors magnify the handiness of the man-machine interface.

This kind of teleoperation diagram is often called *Teleassistance* or *shared control* because sometimes the operator directly controls the remote system and at other times the system is momentarily autonomous with goals to achieve. Making the remote system more autonomous is the common way researchers are working on. It helps the handiness of the whole control and prevents from delay drawbacks. This is the framework we are interested in.

2. OUR APPROACH

2.1 Teleoperation Strategy

We aim at teleoperating a mobile manipulator through the *Internet*. The next paragraph introduces the teleoperation architecture we have implemented and section 3 briefly presents the simulation model and the results we have already obtained.

Section 4 details the experimental set-up and the results we obtained up to now. At least, we present the next results we intend to get.

2.2 Teleoperation Architecture

We will use the term "*Base*" to refer to the host computer that features the distant control and the man-machine interface. The "*Remote System*" will point out the system to be teleoperated. To simplify the reasoning, we will consider a monodimensional system. Figure 1 gives a global view of the teleoperation diagram and the next sections give details on each block.

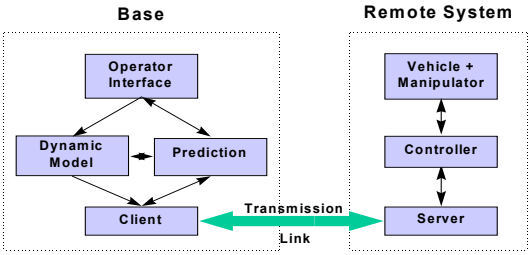


Figure 1 - Summary of the functions of the Base and the Remote System

2.3 The Remote System

The *Remote System* includes the communication with the *Base* (*server block*) and the local control of its state variables: velocity, direction, axes of the manipulator (*controller block*). It also features security systems in case of breakdown or loss of communication.
We will see later in this paper how we improved the whole system by adding a transmission delay compensation system.

2.4 The Base

The *client* block communicates with the remote site through the *transmission link*. The *operator interface* block acts as a virtual reality man-machine interface. It detects and displays security warnings in case of communication trouble. The association of the *prediction* and *dynamic model* blocks allows us to make pre-simulations and display a prediction of the *Remote System* state before receiving it through the transmission link. More details are given in section 3.2.

2.5 Minimum Architecture

We have built such a project using the transmission and *Base* equipment every research laboratory owns. It requires an (even temporary) access to the *Internet* and a bare *PC* computer running *Windows 95/98/NT*. This can be portable to *Unix*-like platforms as we used *C++* programming. A *Java* programming can be envisaged in order to prevent platform incompatibilities.

2.6 The Experimentation Platform

In order to test the whole teleoperation platform, we use a terrestrial vehicle equipped with a 6 degree-of-freedom (d.o.f.) manipulator [7] visible in figure 2. It consists of a terrestrial 6x6 vehicle fitted with a *PUMA* manipulator. Several control laws are featured: global movement of the whole mobile manipulator [8], force-driven control laws, *PID* control law, ... A *dSpace*® *DSP* board is dedicated to the low-level control of the 8 axis (6 for the arm and 2 for the vehicle). A laptop PC fitted with a wireless *2Mbps* Ethernet network board takes care of the transmission and of a global control of the whole. A second *PC* deals with the *GPS* acquisitions and video feedback (2 available webcams). These two *PCs* run *TCP* servers to be connected to the *Base* host.

Figure 2- Picture of the mobile manipulator

3. SIMULATIONS

3.1 Initial Experimentations

In order to build a simulation model, we have performed preliminary experiments [7] using the experimentation platform introduced in section 2.6.

These various experiments have led to a model of the transmission line and of the *Remote System* as a monodimensional 2nd order discrete filter without making strong assumptions. We have also established the stochastic laws concerning the round trip delays of the network communication used for these experiments.

3.2 Simulations

These results allowed us to build a generic monodimensional simulation model [9] whose global organization is presented in figure 3.

Every *500 ms*, the *Base* and the *Remote System* send back and forth messages and acknowledgments through the transmission blocks that randomly delay theses messages. They both are able to measure the time spent from the

emission of a message to the reception of the corresponding acknowledgment.

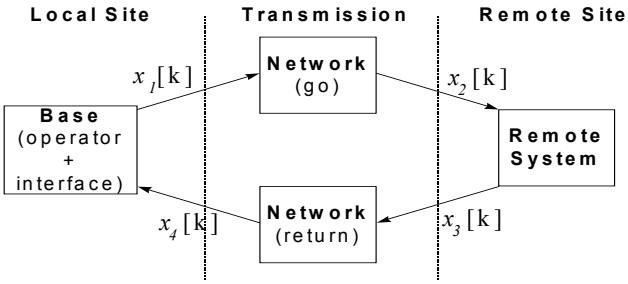


Figure 3 - General organization of the simulation

In figure 4, we can observe the response of the *Remote System* x_3[k] directly excited by steering reference values (x_2[k]).

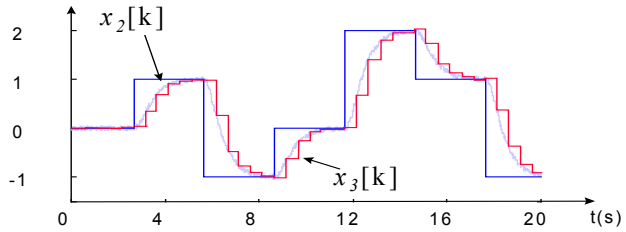


Figure 4 – Response of the Remote System to step values

Globally, the effects of both go and return through the network are visible in figure 5 particularly on x_3[k] : transmission lines don't only delay the signals but they also distort them by making their sampling period vary.

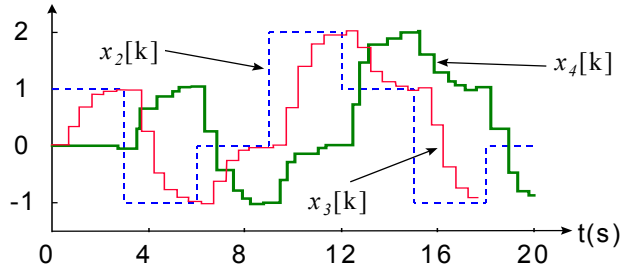


Figure 5- Effects of the network on x_3[k]

In this context, in order to eliminate these variations, we have added two "*Delay Variation Compensators" (D.V.C.*) at the input of the *Base* and the *Remote System*, respectively. They stack incoming samples as they asynchronously arrive; meanwhile they unstack these samples (in the same order : *First-In, First-Out*) with their very initial constant sampling rate. The results are that signals find again the shape they initially had but the mean delay is greater. Figure 6 illustrates the effects of the *DVC* at the input of the *Base*; x_{4c}[k] (x_{4c} stands for *corrected* x_4) is the signal at the output of the *DVC*.

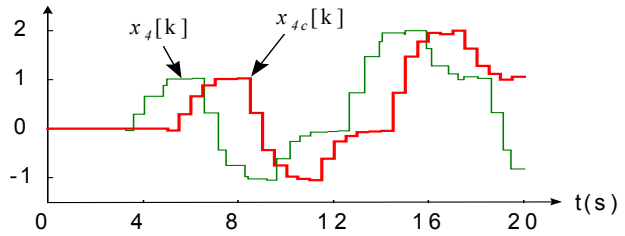


Figure 6 - Effects of the DVC

If we would take a look at the *final round-trip-time (FRTT)* — .i.e. the time spent through the network (once), the *Remote System DVC* (for instance), the network (a 2nd time) and at last the *Base DVC* — we would see it is constant. This allowed us to build a prediction system in the *Base* (in fact, the presence of the *DVCs* is a condition sine qua non to obtain good results with the predictor). This predictor (illustrated in figure 7) uses the reference signal emitted by the *Base* x_1[k], the signal outputted by the *Base DVC* x_{4c}[k] and the *FRTT*. It internally simulates the action of the network (*ERTD* is an estimation of the next *FRTT*) by using a dynamic filter which coefficients are provided by an adaptive filter trying to match its results with the behavior of the *Remote System*.

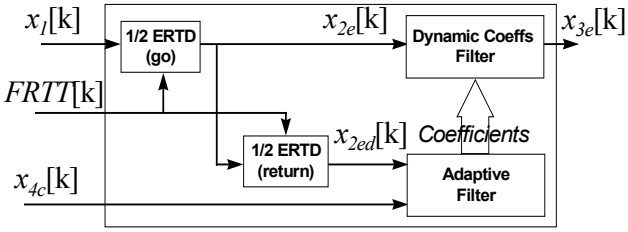


Figure 7 - Remote System state estimation

Figure 8 shows some results of estimation with a reliable permanent pattern. More details are given in [9].

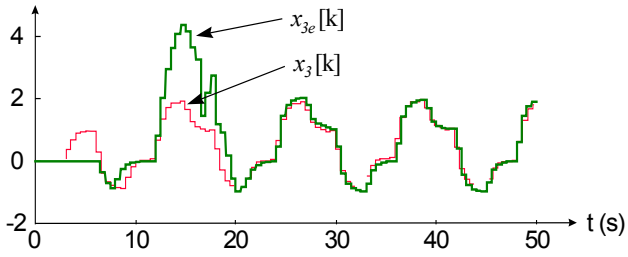


Figure 8 - Remote System real state and estimation

4. EXPERIMENTATIONS

4.1 Introduction

This section will introduce the communication architecture we have set up with the aim of building a real teleoperation experimentation platform. We will observe the quality of the computer environments we use and the results of a real *DVC* in several situations.

4.2 Global Communication Architecture

4.2.1 Communication Between the Base and the Remote System

We have set up two parallel symmetrical *TCP* channels that are simultaneously used as visible in figure 9:

- *BASE* is the application located on the operator's desk (the *Base*).
- *MANIMOB* is the application located on the laptop of the *Remote System*.

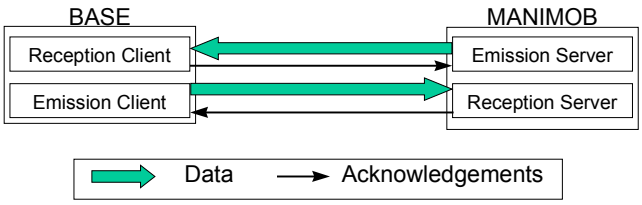


Figure 9 - Global architecture of telecommunication

Both applications take charge of the emission/reception the same way using the plan depicted in figure 10.

The *"Emission"* block deals with the emission of messages to the other computer and also with the reception of acknowledgments the other computer sends.

The *"Reception"* block takes care of the reception of messages sent from the other computer and also of sending back acknowledgments.

Next section details the layers functioning.

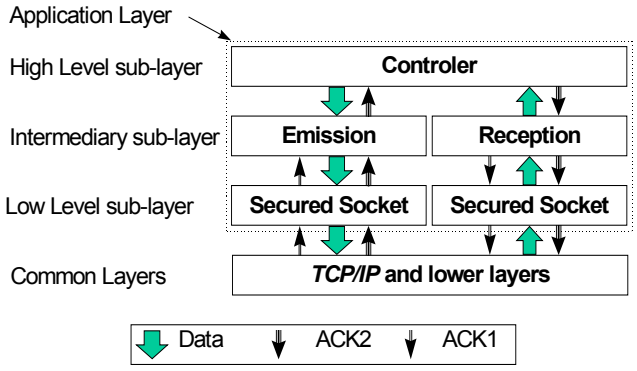


Figure 10 - Architecture of telecommunication parts of both applications client and server

4.2.2 The Low-Level Sub-Layer: Secured Socket

This sub-layer is located at the lower level of the application layer.

It retrieves in one hand the data received by the client/server socket in order to retrieve the real message and to check it. If the check is successful, the layer sends it to its top sub-layer.

In the other hand, this sub-layer intercepts the data that its top sub-layer transmits to it and it encapsulates the message in a frame including extra data enabling to check the data at its arrival.

A frame is made this way (cf. figure 11):

- SOH : Start of Header; a byte that allows this sub-layer to check whether we are reading the beginning of a frame and to re-synchronize to the next packet in case of problem.
- CHK : a checksum that verifies the validity of the data.
- LEN: length of the data (in bytes).

1 byte	*1 byte*	*2 bytes*	*n bytes*
SOH	**CHK**	**LEN**	**DATA**

Figure 11 - Format of any exchanged frames

4.2.3 The Intermediary Sub-Layer

This layer is dedicated to the network timing measurements. Sections 4.2.3.1 and 4.2.3.2 detail the way we operate.

4.2.3.1 Emission Block

Every message sent by the client/server application is encapsulated in a frame that includes extra timing data (cf. figure 12), for instance: `[4850]TEST`

[	*Emission Date in ms (ASCII)*	]	*Data*

Figure 12 - Format of frames sent to the low-level layer

The emission date is the number of milliseconds spent since the beginning of the connection.

This part also manages the acknowledgment messages sent by the opposite reception block. As explained in the next chapter, this frame includes the primary emission date. It is so possible to compute the network round-trip-time (*NRTT*) by differentiating the reception date with the primary emission date.

The *NRTT* is used by the applications to calculate the ideal dimensions of the compensation queues of the *DVCs* (see section 4.5).

4.2.3.2 Reception Block

This block receives decoded frames emitted by the opposite application from the low-level sub-layer. It sends back an acknowledgment by copying the primary emission date included in the frame it has just received and by sending a *"ACK1"* message as visible in figure 13. This allows the emission part to compute the network round-trip-time *NRTT*.

[	*primary emission date in ms (ASCII)*	]	ACK1

Figure 13 - Reception Block Acknowledgments

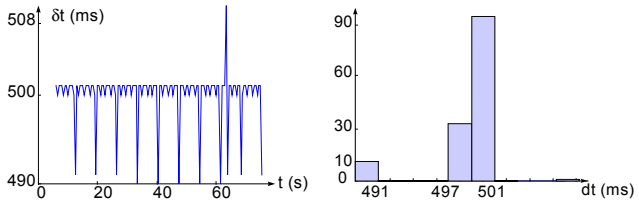
4.3 Real Time Environment

The quality of the real time environment we are using is important to characterize: we won't be able to obtain very good results if the clock timings are not accurate.

We use *Windows 95* for the laptop running *MANIMOB* and *Windows NT* for the computer running *BASE*. The sampling period has been arbitrarily set to *500 ms*. We needed to call the more regularly the control functions every sampling period; therefore, we have used the timer supplied by the *Windows API*. However, the sampling clock this timer provides isn't accurate enough: according to the time spent in the control functions, the periods are not constant and it globally drifts in time. To limit the variations of period and to avoid a time-drift, we compute the next best sample date each sampling time, taking account of the duration of the control functions.

We have analyzed the results of this method; figures 14 (a) and (b) show some real periods we obtained. The regular drops are due to a periodic activity of *Windows*. They are limited to *+/- 10 ms*, i.e. *2%* of the sampling period. This is to be taken into account in the quality of the future results. The distribution of these periods shows a mean period of 500,0 ms and a standard deviation of 3 ms. Figure 14 (c) and (d) illustrate the difference Δt between the sampling dates we obtained and a virtual ideal clock (based on the clock of the computer). We can notice the absence of time-drift (no global slope). The distribution of these differences gives a standard deviation of *3 ms*.

(a) Periods of the clock: δt *(b) Distribution of δt*



(c) Drift Δt of the clock *(d) Distribution of Δt*

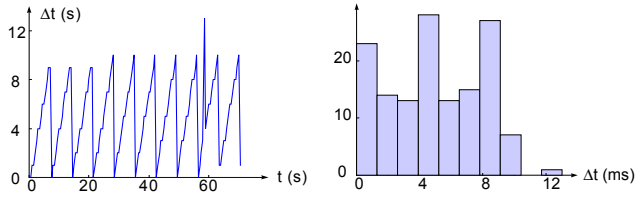


Figure 14 - Quality of the sampling clock

4.4 Simulating a Large Network

For a main part of the experiments, we have simulated a large network by inserting a third application (which name is *RELAY*) that just delays every frame transmitted between *BASE* and *MANIMOB* (see figure 15). Each frame is delayed of a value computed according to:

- a stochastic law we can choose (*Gauss* or *Poisson*),
- the parameters of the chosen stochastic law (mean value, standard deviation, or μ),
- on each channel, the frames go out in the same order as they arrived; as if they were just delayed by a long distance connection.

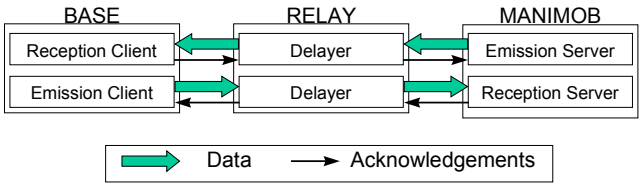
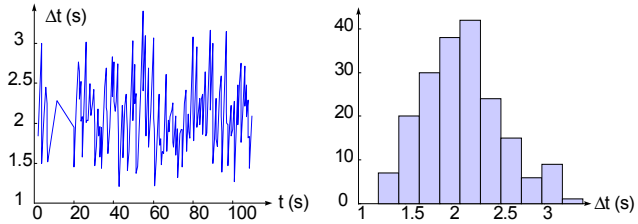


Figure 15 - Global organization with DELAY

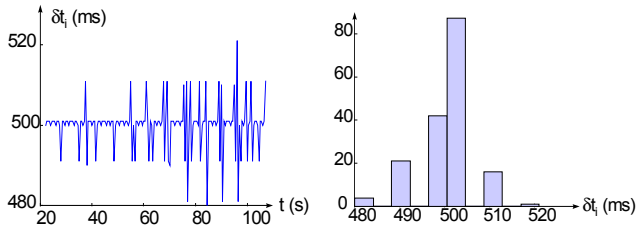
We are thus able to test the algorithms in many different situations as we need.

We have checked the results of *RELAY* by capturing the frames that go in and out of it. Figure 16 (a) and (b) represent the real time spent in *RELAY* for a desired mean delay of *2 s* and a standard deviation of *500 ms*. We really obtain a mean of *2,11 s* and *435 ms* of standard deviation for a *2 min* long capture. Subfigures (c)-(d), (e)-(f) give an idea of the action of *RELAY* on the in/out frame periods. Subfigure (g) shows in and out frames dates.

(a) time spent in RELAY: Δt *(b) Distribution of Δt*



(c) Periods of input frames δt_i *(d) Distribution of δt_i*



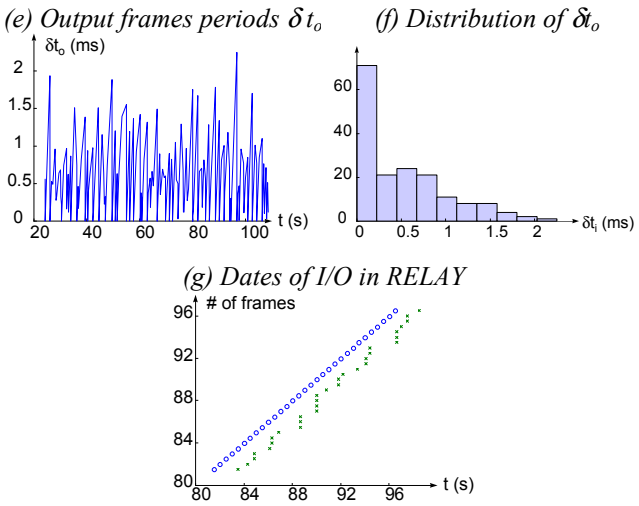


Figure 16 - Obtained Delay Values

4.5 Delay Variations Compensator

4.5.1 How it Works

On both side (client and server), we have inserted a *Delay Variations Compensator (DVC)* as visible in figure 17. It is simply a queue that stacks the messages asynchronously arriving and it synchronously un-stacks them every sample period.

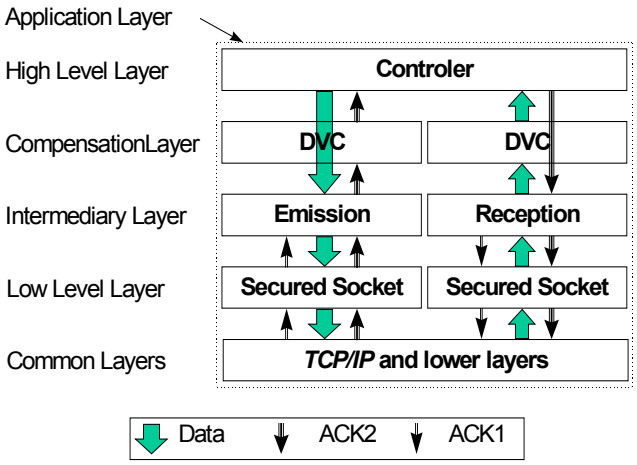


Figure 17 - Insertion of the DVC in the transmission chains

There is an initialization pattern at the beginning in order to set up the best dimension of the queue :

- about ten test frames are sent from both *BASE* and *MANIMOB*.
- Every *Network Round-Trip-Time (NRTT)* is then analyzed. We use the amplitude of the variations of these *NRTT* to set the dimension of the *DVC* queue, with a factor of security.

This factor of security prevents to empty the queue whenever the network becomes slower or more unstable. A security rule, used to prevent the emptiness of a *DVC*, makes the system go into a transient test mode in order to set a new best size for the *DVCs* as soon as it goes under a certain threshold. Meanwhile, the control of the *Remote System* is paused to prevent any accident; it runs again when the *DVC* goes in its new steady pattern.

We also use another timing variable called *Final Round-Trip-Time (FRTT)*. In this variable, the time spent in the *DVCs* is added to the time spent through the network. Figure 18 illustrates the difference between *NRTT* and *FRTT*.

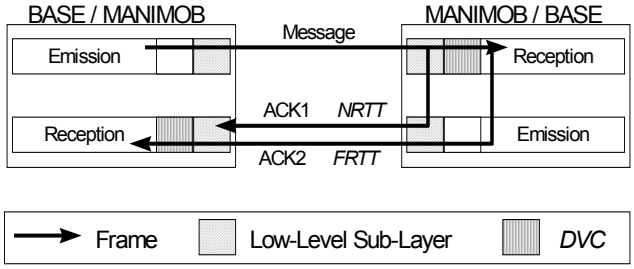


Figure 18 - Difference between NRTT and FRTT

4.5.2 Experimental Results at Very Short Distance

We have first experimented the *DVC* without using any virtual or real strong delay. The computer running *BASE* was located in the same *Ethernet* network as the laptop running *MANIMOB*, taking account of the *2Mbps* radio network connecting the laptop to the local network.

Figure 19 shows different aspects of the regulation; results appear for *Base DVC*:

- subfigure (a) gives an idea of the periodicity of the incoming and outgoing samples. The *DVC* needs to delay every sample just by a little less than *1 s* to re-synchronize them. We can observe that the amount of items in the queue of the *DVC* (subfigure (b)) has stayed stable and it keeps the messages during *650* to *850 ms* (subfigure (c)).
- Statistics and the subfigures (d)-(g) and (e)-(h) of Figure 19 show a better periodicity : the *DVC* has brought back the mean from *500,7 ms* to *499,9 ms* and it has divided the standard deviation of the periods by a little less than *4*: *8,2 ms* versus *30,5 ms* for the incoming messages.
- Subfigures (f)-(i) give an idea of the difference between the output samples dates and a virtual clock; we can observe no drift.

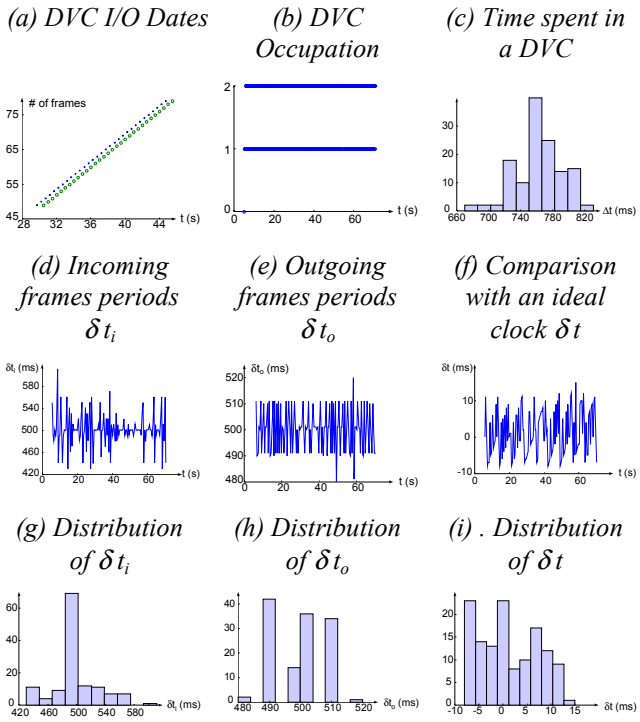


Figure 19 - Results without extra delay

At last, figure 20 allows to compare the *Network Round-Trip-Time (NRTT)* (cf. subfigures (a) and (c)) with the *Full Round-Trip-Time (FRTT)* (cf. subfigures (b) and (d)). For this latter, the transient pattern corresponds to the initialization period, before the *DVC* runs. Steady pattern statistics give a *FRTT* to *1,50 s* (vs. *58,6 ms* for the *NRTT*) with a standard deviation of *7 ms* (vs. *15 ms*).

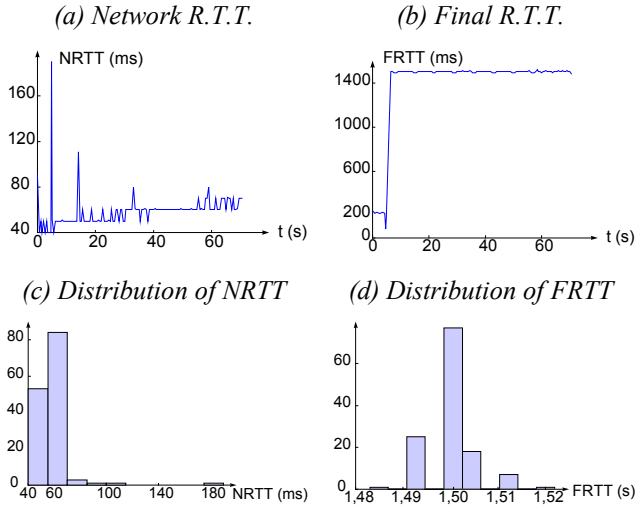


Figure 20 - Incidence on RTTs

To put it in a nutshell, even for little varying delays, the *DVC* increases the quality of the incoming signals. In this case, the mean *FRTT* is three times as much as the *NRTT*; it seems a big ratio but it is due to the combination of the sampling time of *500 ms* (ten times as much as the mean *NRTT* in this case!) with the security factor of the *DVC* .

4.5.3 Experimental Results at Long Distance

In a second time, we have experimented the *DVC* using *RELAY* which mean delay and standard deviation equal respectively to *2 s* and *500 ms*. The different applications have run on the same computers as before.

Figure 21 gives an idea of the different results:

- in subfigure (a), we can observe the quality increase in making the samples more periodic between the input and the output of the *DVC*.
- Subfigures (d)-(g) and (e)-(h) allow us to qualify better this improvement: at input, periods mean and standard deviation are respectively equal to *474 ms* and *437 ms*; at output they respectively equal to *500 ms* and *8,2 ms* (this makes a ratio of 53).
- We can observe in subfigure (b) that the occupation in the queue of the *DVC* has varied from *2* to *7* items in permanent pattern. As the delay variations has not grown during this experimentation, the *DVC* never empties. Subfigure (c) shows a time of presence located between *1* and *4 s* with a mean value of *2,68 s*.

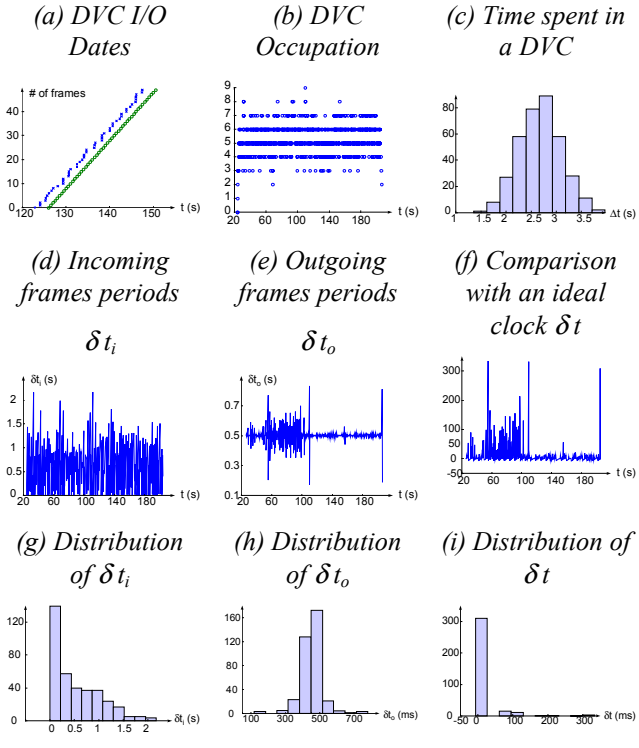


Figure 21 - Results with an extra delay of 2 s

At last, in figure 22, we can see the difference between the *NRTT* (cf. subfigures (a)-(c)) whose mean value and standard deviation are equal to *4,6 s* and *580* ms, respectively, and the *FRTT* (cf. subfigures (b)-(d)) with *9,5 s* and *58 ms* in steady pattern, respectively.

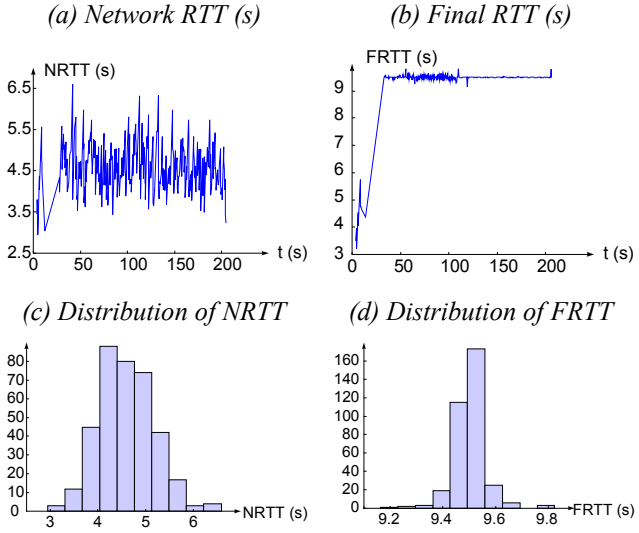


Figure 22 - Incidence on RTTs

To conclude, as long as the delays variations don't grow up to the point to empty the *DVCs*, the results seem good enough to be able to use a prediction system on corrected signals.

Better results may be obtained by using a real-time environment and faster computers ; the lack of quality of the periods at the output of the *DVC* make the predictor more unstable or slow in falling into steady pattern.

4.6 Experiments to be Held

Now that the *DVC*s work, we are able to use the corrected signals they output in order to build a prediction system to simulate what the *Remote System* does before receiving the real data about its status. This is what we are working on; unfortunately, we don't have yet results to be presented in a paper.

5. CONCLUSION

Despite the non real-time environments we are using, the results about the *DVC*s seem encouraging and to be a good base for the building of a predictor which would cope a little with high delays and hence would help the operator in his teleoperation task.

We are currently running experiments to set up the prediction part we had simulated. In parallel we are working on the integration of *GPS* acquisitions and video feedback in the teleoperation platform.

In those experiments, the information perception from the vehicle can be improved by the adjunction of different transducers as accelerometers, magnetic compass, force sensors, CCD cameras, laser telemeters and ultrasonic sensors. The simulator will then be more accurate since it will receive richer information and the operator will be able to view simultaneously both the simulation and the reality on his screen.

6. REFERENCES

[1] C. P. Sayers, D. R. Yoerger, R. P. Paul, J. S. Lisiewicz, *"A Manipulator Work Package for Teleoperation from Unmanned Untethered Vehicles",* Proc. of the IARP Workshop on Subsea Robotics, Toulon, France, 1996.

[2] G. Hirzinger, K. Landzettel, CH. Fagerer, *"Telerobotics with large time delays - the ROTEX experience"*, Proc. of the Intl. Conf. on Intelligent Robots and Systems (IROS' 94), Munich, Germany, 1994, pp. 571-578.

[3] K. Taylor, B. Dalton, *"Issues in Internet Telerobotics"*, Proc. of the Intl. Conf. on Field & Service Robotics (FSR'97), Canberra, Australia, 1997.

[4] P. K. Pool, D. H. Ballard *"Remote Teleassistance"*, Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA'95), Nagoya, Aichi, Japan, 1995, pp. 944-949.

[5] H. Turchi, A. Crosnier, P. Fraisse, *"Realtime Environment for Mission Programming of Telerobotics Systems"*, Proc. of the SPIE Conf. on Intelligent Systems, Pittsburgh, USA, 1997, pp. 22-24.

[6] A. Rastogi, P. Migram *"Augmented Telerobotic Control"*, University of Toronto, Canada.

[7] A. Lelevé, P. Fraisse, A. Crosnier, P. Dauchez, F. Pierrot, *"Towards Virtual Control of Mobile Manipulators"*, Proc. of the 3rd World Automation Congress (WAC'98), Anchorage, USA, 1998.

[8] C. Perrier, P. Dauchez, F. Pierrot, *"A Global Approach for Motion Generation of Non-Holonomic Mobile Manipulators"*, Proc. of the IEEE Intl. Conf. On Robotics and Automation (ICRA'98), Leuven, Belgium, 1998, pp. 2971-2976.

[9] A. Lelevé, P. Fraisse, P. Dauchez & F. Pierrot, *"Modeling and Simulation of Robotic Tasks Teleoperated through the Internet",* Proc of the Intl. Conf. on Advanced Intelligent Mechatronics (AIM'99), Atlanta, USA, 1999.