End-to-end congestion control protocols for remote programming of robots, using heterogeneous networks: A comparative analysis

Raul Wirz , Raul Marín , José M. Claver , Manuel Ferre , Rafael Aracil , Josep Fernández

Computer Engineering and Science, University Jaume I (UJI), 12071 Castellón, Spain
Dept. de Informàtica, Universitat de València, 46100-Burjassot, Spain
Automatics, Electronics Engineering, and Industrial Computers Department, Universidad Politécnica de Madrid, E-28006 Madrid, Spain
Computer Science Department, Universidad Politécnica de Cataluña, Spain

Keywords:
Networked robots
Internet congestion control protocol
Telerobotics
E-learning
Industrial robotics telelaboratory

A B S T R A C T

There are many interesting aspects of Internet Telerobotics within the network robotics context, such as variable bandwidth and time-delays. Some of these aspects have been treated in the literature from the control point of view. Moreover, only a little work is related to the way Internet protocols can help to minimize the effect of delay and bandwidth fluctuation on network robotics. In this paper, we present the capabilities of TCP, UDP, TCP Las Vegas, TEAR, and Trinomial protocols, when performing a remote experiment within a network robotics application, the UJI Industrial Telelaboratory. Comparative analysis is presented through simulations within the NS2 platform. Results show how these protocols perform in two significant situations within the network robotics context, using heterogeneous wired networks: (1) an asymmetric network when controlling the system through a ADSL connection, and (2) a symmetric network using the system on Campus. Conclusions show a set of characteristics the authors of this paper consider very important when designing an End-to-End Congestion Control transport protocol for Internet Telerobotics.

1. Introduction

One of the multiple applications of Networked Robotics is enabling Internet access to expensive devices (e.g. industrial robots, FPGA systems, conveyor belts, etc.) organized as a telelaboratory for education. Thus, students and researchers can program their own robotic experiments via Internet, and then obtain the results through, for example, a simple webpage

One essential part of a Telelaboratory is the interconnection of sensors, cameras, and robots via a networked system In the scientific literature, much work can be found that propose different ways and architectures to organize task-oriented applications of multiple network robots Some of these architectures are focused on Internet software frameworks (e.g. Web Services at the application OSI layer), and have been extended from previous work in single-robot telerobotics.

Other work focuses not only on application protocols, but also on other levels of the OSI layers, like transport and network, which enable real-time control and teleoperation of network robots over IP. In fact, solutions can be found to cope with the problems associated with the Internet, in order to control networked robots: (1) time-varying transmission delay, and (2) non-guaranteed bandwidth.

First of all, in this paper we present the IP-based network architecture of the UJI Industrial Telelaboratory (see Figs. 1 and 2). After that, some of the most recent approaches of Network Robot Control under time delays are presented, which offer some interesting solutions designed to guarantee telerobotic system stability, when the Internet is used as the medium of communication. Then, we will focus on the transport protocols that enable end-to-end congestion control in a TCP-Friendly manner for teleoperation, and tele-programming of robot arms. Simulations using TCP, UDP, trinomial TEAR (TCP Emulation at Receivers), and TCP Las Vegas protocols are presented within the UJI Industrial Telelaboratory. In fact, two different situations are studied: (1) using an asymmetric network (i.e. user controlling the devices through a ADSL connection), and (2) a symmetric network (i.e. on campus). Then, from these results, a set of conclusions are obtained which are important in order to design an end-to-end congestion control transport protocol for Internet teleoperation.

2. The UJI industrial telelaboratory network architecture

In Fig. 2, we can see the Network connectivity of the UJI Industrial Telelaboratory. In fact, in this system we consider that

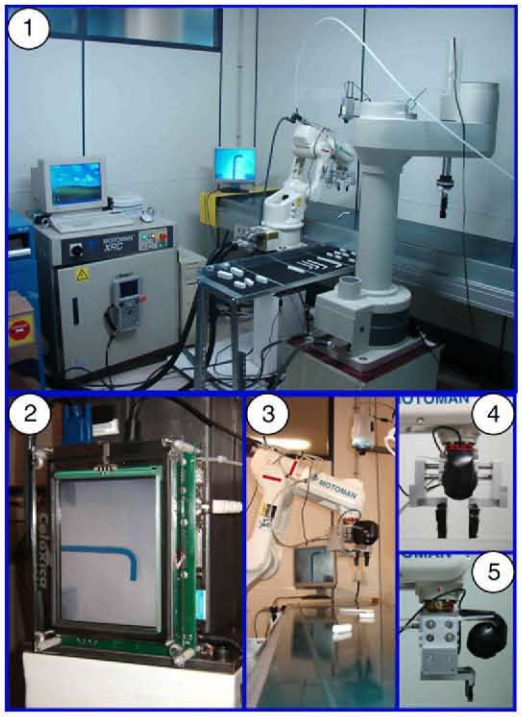

Fig. 1. The UJI Industrial Telelab devices: (1) Motoman industrial manipulator, (2) FPGA based vision system, (3) Conveyor belt and monitoring camera, (4, 5) On-hand mounted camera.

every device (i.e. industrial robot, conveyor belt, FPGA, etc.), is connected to the same Ethernet network, and they act as single Network Robots that communicate with each other through the SNRP application level protocol (A new application level protocol designed by our team,). This architecture offers many advantages, like scalability and maintainability, and it introduces interesting issues, like device synchronization.

In order to make the SNRP simple to use and implement, it uses HTTP protocol as a basis, which give it more interoperability and flexibility. However, for this kind of situation, HTTP does not provide the following features: (1) Event Notification, and (2) Support for structured information. These two characteristics are very important to design the SNRP framework in industrial robotics. To accomplish this, we have incorporated the REST model into the SNRP protocol, which permits the implementation of state-oriented applications and a simple scenario to design event notification, and structured information features.

Simplicity is maybe the most important challenge of network robotics architecture, due to the fact that it must be possible for a very broad range of devices to be part of it. In fact, as explained in [11], thanks to this simplicity, we were able to implement a prototype of SNRP Network Camera using a FPGA.

First of all, as we want to enable the devices to be accessed through the Internet, they should be able to manage IP protocol. On top of that, the SNRP framework enables the device to accept TCP, and UDP connections. As explained in Section 4, UDP and TCP are not the best solutions to perform remote control through the Internet, so the SNRP framework is being designed to provide the possibility of transporting Internet packets through other transport protocols, like Trinomial, TCP Las Vegas, TCP Reno, or TEAR (see Fig. 3).

3. Network robot control under time delays

The Internet is a suitable way for developing the communication channel of a remotely controlled robot. However, some points

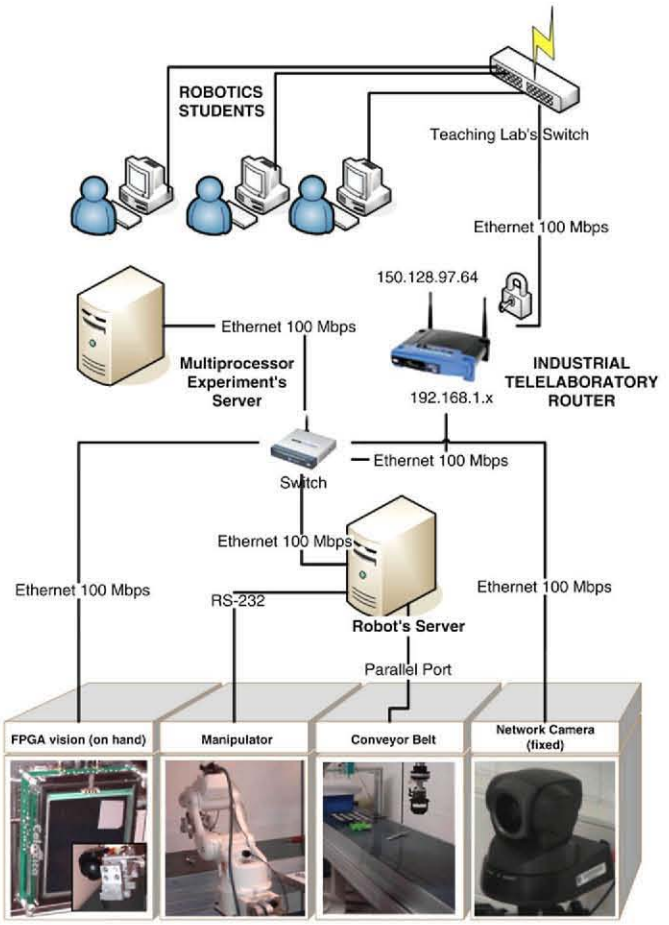

Fig. 2. The UJI Industrial Telelab networking configuration.

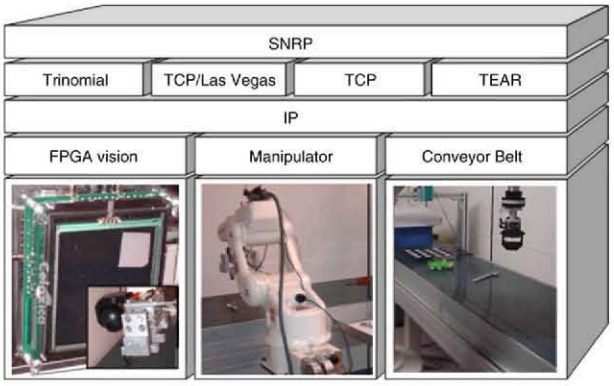

Fig. 3. The SNRP framework.

must be taken into account from the control point of view, such as reliability, time delay and bandwidth. Therefore, a communication protocol has to be selected according to these parameters.

Two kinds of commands have to be considered to control a remote robot: high level commands and low level commands. High level commands are used when user is sending commands related to the task, such as 'get a part', 'move to home position', 'close the grip', etc. Transmission of these commands must be guaranteed in order to properly execute remote tasks.

Commands are generated according to the task execution procedure; therefore low frequencies are required. In this case, TCP is used, since it is a reliable protocol, and packets are retransmitted when they are lost or corrupted

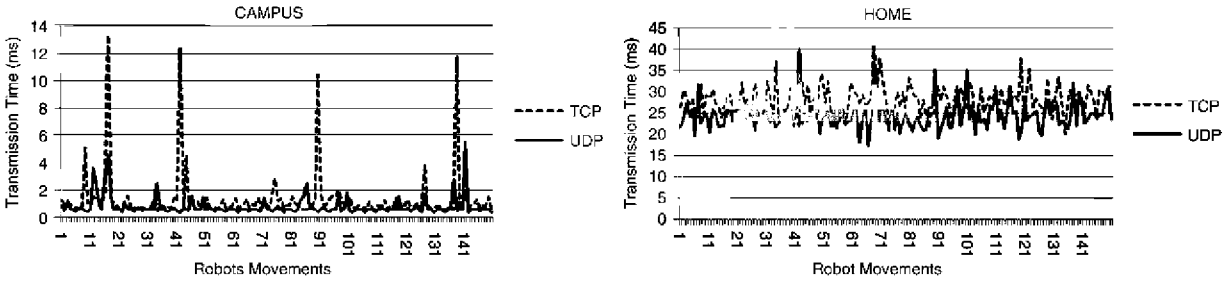

Fig. 4. Delay response when controlling an industrial Motoman robot via internet using UDP and TCP.

On the other hand, low level commands have different requirements. This kind of command is related to robot movement. It implies a stronger connection between user and robot, such as guiding a manipulator or a mobile platform. In this case, higher bandwidth is required, but reliability is not a critical factor. UDP protocol is usually used for these tasks, since packets are minimized and delays are reduced If some packets are lost, then the remote robot can continue working, but with poorer performance.

An interesting example for the use of low level commands, is master–slave teleoperation with force feedback. In this case, two data flows run continuously. First, the user handles a device called master that generates movement references for the robot (slave device); second, interaction forces between robot and environment are retransmitted to the user via force reflection in the master device. These systems are called bilateral, and are very sensitive to communication time delays Passive control techniques and scattering variable transformation are applied in order to guarantee stability of bilateral systems, in the presence of significant communication time delays

Control problems increase when a master–slave system is linked via the Internet, since some data can be lost, and communication time delay is variable. Discrete scattering techniques are used to implement a switching packet transmission line that is also passive when communication delay is variable, and some packets are lost Several passivity based strategies have been proposed to passively control master and slave sides. A generic framework for geometric telemanipulation of port-Hamiltonian systems has been proposed in

4. Transport protocols for remote control of network robots

The basic transport protocols available in the Internet for implementing remote control applications are the following:

1. *UDP* (User Datagram Protocol) This is based on the idea of sending a datagram from a device to another as fast as possible (i.e. best effort). This protocol does not guarantee that the information will reach the destination, and besides this, it does not manage any network congestion situation.
2. *TCP* (Transmission Control Protocol) This guarantees the application level that the information will reach the destination performing the necessary retransmissions. Moreover, TCP takes care of network congestion and adjusts the transmission accordingly.

UDP is a protocol that does not maintain a connection with the Server side, and it does not retransmit lost packets, it does not control network congestion, and neither manages any confirmation of packets that have reached their destination. The advantage of UDP, for remote control of devices via the Internet, is that having good network conditions, communication is accomplished without significant delay and without important fluctuations (i.e. delay jitter). Moreover, UDP does not assure that the packets have reached the destination in the proper order that they were sent; if fact, UDP does not inform if packets have even been received, or not. Besides this, UDP does not perform any congestion control mechanism, which means the sending rate is not adapted according to the real bandwidth available. This situation implies that we need another protocol for remotely controlling devices via the Internet.

On the other hand, TCP is a very sophisticated protocol that establishes a virtual connection between the sender and the receiver. Moreover, as TCP manages the confirmation of packets received properly, we can assure that communication will be reliable. However, when TCP was designed they had in mind reliable communication for applications such as e-mails and files (ftp), and not controlling devices such as robots. The congestion control mechanism and connection establishment imply having high delay jitter (fluctuation), a situation that is not appropriate for applications such as Internet teleoperation of a robot manipulator using a haptic device. In the Fig. 4 we can see the results obtained when controlling a real robot using both, TCP and UDP.

The majority of current telerobotic applications using the Internet (e.g. telelaboratories) use TCP or UDP. For this, the variable time-delay and bandwidth effect is resolved in the application level, by using intelligent sensors, predictive displays, and high level commands. On the other hand, if we really need to perform a teleoperation, we need to find applications that are closer to real time In this situation we need more specific communication protocols

As this is a very emergent research field, in the scientific literature we cannot find many articles describing specific protocols to teleoperate networked devices (i.e. like robots) via Internet. On the other hand, we can find many protocols to design networked applications that require the transmissions of Multimedia content via Internet: (1) TFRC (TCP-Friendly Rate Control Protocol) RAP (Rate Based Adaptation Protocol) LDA (Loss-Delay Adjustment Protocol) SIMD (Square-Increase/Multiplicative-Decrease Protocol) and RTP (Real Time Protocol) These protocols are not very convenient for telerobotics due to the fact that they use an intermediate buffer to compensate the delay jitter when receiving video and audio. In telerobotics using buffers implies obtaining an overall higher delay that seriously reduces the smoothness with which the robot can be controlled.

Some of the few works that specifically design communication protocols for Internet teleoperation are the following:

(1) *Trinomial method* : It is a rated-based protocol, which means it manages the network congestion by adjusting the inter-packet gap (IPG) instead of the window size schema that uses TCP. Thus, the protocol controls the number of datagrams per second depending on the available bandwidth. The Trinomial method uses UDP as basis. It means that the Trinomial is able to adapt to the network congestion and available bandwidth without affecting very much the way the user teleoperates the robot. As observed in [5], the Trinomial protocol provides a sending curve that is quite smooth and makes better use of the available bandwidth, thus

obtaining a very good efficiency compared to the UDP and TCP protocols. In the following section we will study some parts of the Trinomial that we consider can be improved in order to be applied in the telelaboratories field.

(2) *Real-Time Network Protocol (RTNP)* : is specially designed for bilateral teleoperation using mater/slave manipulators and force feedback. In such a system, the time-delay can be produced by the performance of network devices (i.e. routers, switches, etc.), the end-to-end congestion control algorithms, or the implementation of the network stack in the hosts. This is a protocol that uses an identification in the UPD/TCP headers to inform the Linux-based real-time operating system that the received packet has the category of "real time", in order to give it the maximum priority when passing the packet to the application level. The RTNP shows that the overall time-delay between the client and the server depends not only on the network but also on the software provided by the operating system. The RTNP focuses on the network stack implementation on the hosts instead of studying end-to-end congestion control techniques, which is the subject of this paper. This is why this protocol is not included in the network experiments.

(3) *Interactive Real-Time Protocol* (IRTP) : is an IP-based protocol that takes the advantages of both, TCP and UDP, to improve the response in teleoperation systems. It is a connection-oriented protocol that implements congestion control and error control. To enhance the efficiency, the IRTP protocol simplifies the packet header as much as possible, so that a major relationship between the data that is sent by the application level and the protocol control information is obtained. Moreover, the IRTP reconfigures itself in order to transmit the two basic kinds of data that are transmitted in a network control system, which are: (1) the crucial data (i.e. information that must reach the destination even if it has some time delay), and (2) the real-time data (i.e. information that must reach the destination as soon as possible). The IRTP protocol uses the same control congestion algorithm as the Trinomial method. As we already have the Trinomial protocol included in the results, we have not performed the experiments with the IRTP protocol.

Moreover, in the telelaboratories context there are situations where the student/researcher is performing an experiment from home using an ordinary ADSL connection. This kind of asymmetric communication normally gives a poor upload link and a good download bandwidth. The TEAR protocol (TCP Emulation at Receivers) [7] is specifically designed for the transmission of multimedia streams on asymmetric connections. The TEAR protocol does not perform retransmission of lost and corrupted packets. Moreover, it does not use an ACK for every packet that has been sent. So that, when we speak of RTT in TEAR we are referring to the last packet received in addition to the ACK used by the TEAR after sending that packet. The following sections will provide some simulations to compare the performance of the trinomial, TCP, TCP Las Vegas, and TEAR protocols when performing an experiment within the telelaboratory. The experiments present two situations: (1) using symmetric network on campus, and (2) using an asymmetric network including a user connection to the network robots from home via an ADSL connection.

5. Experiment description

In this section we are going to study the behaviour of the TCP, TEAR, Trinomial, and "TCP Las Vegas" protocols for a remote visual servoing experiment performed by a student with both, at home using an asymmetric ADSL connection to the telelaboratory (i.e. 320 Kbps-Upload and 1Mbps-Download bandwidth), and on campus using the symmetric Ethernet structure at 100 Mbps. For this, the student asks the telelaboratory to provide as much

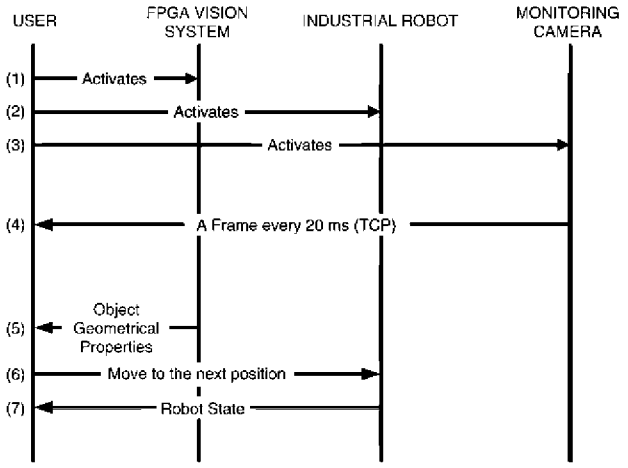

Fig. 5. Experiment data flow.

information from the FGPA-Vision System as possible, and he/she performs the control algorithm to provide the next position of the robot. The control Law is calculated by the student in his own computer. Moreover, in these simulations the student is provided with a packet from the monitoring camera every 20 ms, using a TCP Reno flow.

As we can see in Figs. 5 and 6, the user (i.e. Node 10 at home and Node 3 on Campus) performs a visual servoing experiment over the industrial telelaboratory. These students use the following communication flows:

1. The user activates the FPGA Vision System.
2. The user activates the Industrial Robot.
3. The user activates the Monitoring Camera.
4. The user receives a frame from the monitoring camera every 20 ms, using the TCP protocol.
5. The FPGA Vision System sends the Object Geometrical properties to the user.
6. The User calculates the control Law and sends the next robot position. The robot needs the commands to reach its controller with a minimum gap of two milliseconds. Otherwise the robot would indicate that an error has occurred within its controller. The robot returns its state to the user.

In the simulation, the student gets the object geometrical properties in camera coordinates from the FPGA (e.g. grasping line). From this, the student applies a control law following the on-hand visual servoing control until the grasping line is centered at the middle of the gripper.

6. Results using an asymmetric network (at home)

In this section we are going to observe the RTT behaviour and the bandwidth of Trinomial, TCP, TCP Las Vegas, and TEAR protocols for the industrial telelaboratory using an asymmetric network.

As seen in Fig. 6, we have Node 4 that represents the industrial robot of the telelaboratory. Node 7, represents the router that gives access to every device in the telelaboratory. Node 3 represents a student that is connected to the telelaboratory and he is monitoring the experiment performed by node 10. Node 10 represents a student that is performing a teleoperation (or visual servoing) experiment on the industrial robot (i.e. node 4). In the simulation, traffic to Node 3 is TCP based and it does not generate congestion to the intermediate network routers, because they use a 100 Mb/s network, and the available bandwidth is enough for the whole experiment. Moreover, traffic to Node 3 does not affect the one that goes to Node 10. The traffic from Node 10 (i.e. the

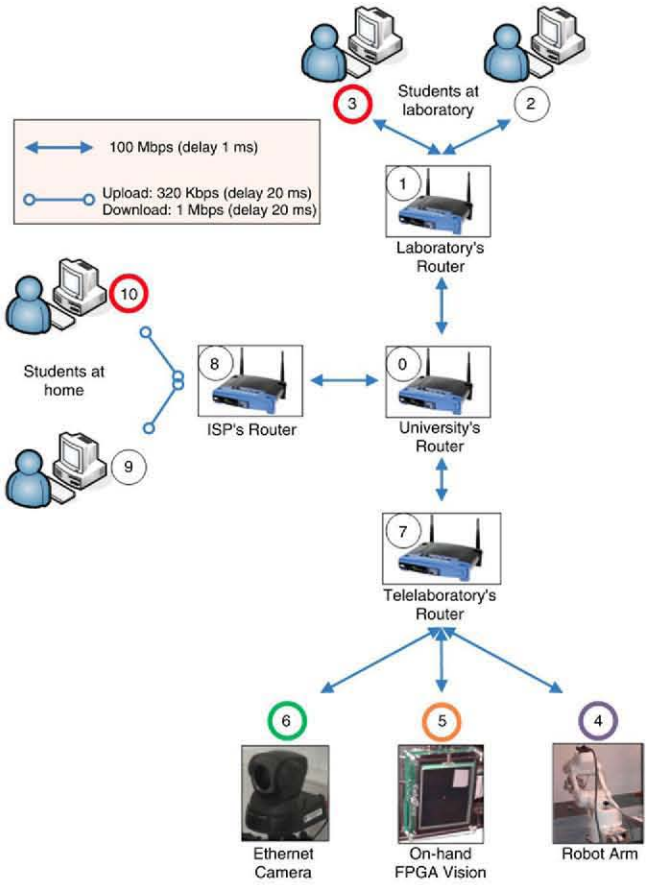

Fig. 6. Nodes configuration for the NS-2 simulations.

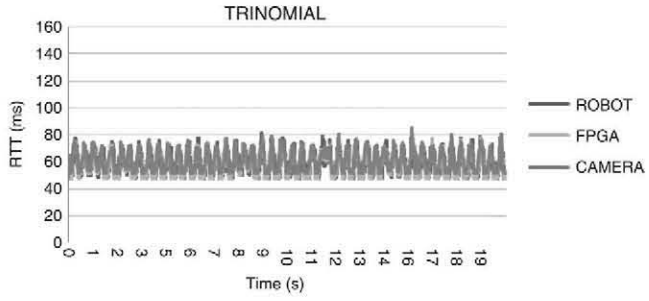

Fig. 7. Results of the RTT behaviour NS-2 simulation when Node 10 uses the Trinomial protocol.

experiment) will vary from Trinomial, TCP Reno, TCP Las Vegas, and TEAR.

As we can observe from Figs. 7–10 and Tables 1 and 2, the Trinomial protocol almost consumes the entire available bandwidth (see Fig. 12) at the router, obtaining an average RTT of 58.45 ms. Moreover, there are packets that almost reach 90 ms of RTT. The Trinomial protocol sets the router buffers to the maximum load, which implies increasing the RTT average between the student and the robot. On the other hand, the Trinomial protocol sends more packets per second than TCP, increasing the information that comes from the student to the robot, and vice versa. Moreover, the trinomial loses almost the 20% of the packets that are sent, which is one of the most significant problems we found with this protocol.

As shown in Figs. 9 and 10, the TEAR protocol is smoother than the TCP, which is very appropriate for the transmission of control information (i.e. robot and FPGA). Moreover, it allows the user to send more control packets to the robot in less RTT. The TCP Las

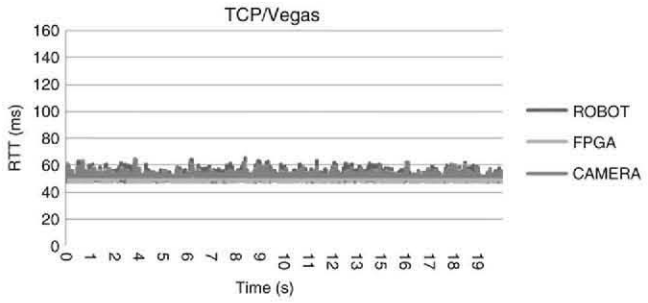

Fig. 8. Results of the RTT behaviour NS-2 simulation when Node 10 uses the TCP Las Vegas protocol.

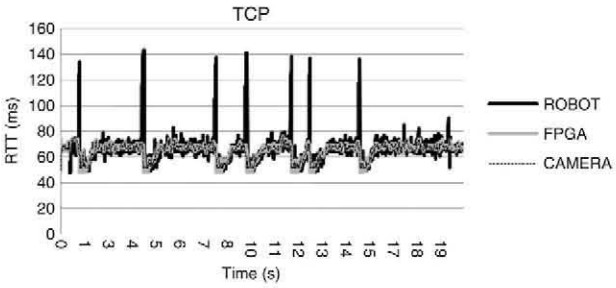

Fig. 9. Results of the RTT behaviour NS-2 simulation when Node 10 uses the TCP protocol.

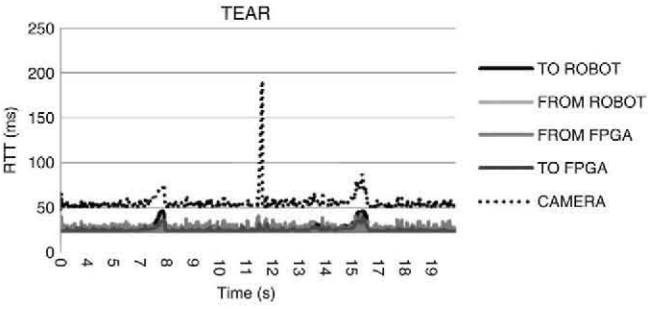

Fig. 10. Results of the RTT behaviour NS-2 simulation when Node 4 uses the TEAR protocol.

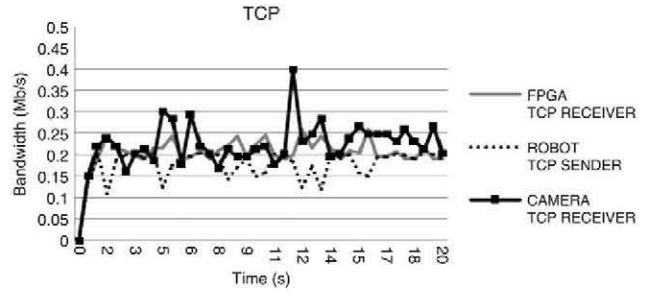

Fig. 11. Telelaboratory experiment using TCP for the Robot, the FPGA, and the Camera.

Vegas also presents many interesting features like the RTT stability, but it does not perform as the TEAR protocol using asymmetric networks.

From the bandwidth point of view, the TCP protocol consumes 80% of the available bandwidth (see Fig. 11), at an, with an average RTT of 66.95 ms. On the other hand, as TCP performs retransmissions, the number of received packets at Node 0 is not as significant as using the Trinomial protocol.

The TEAR protocol is the one that sends more packets to the robot, taking advantage of the asymmetric network configuration.

Table 1
Number of packets sent/received/dropped per flow and protocol in asymmetric simulation

FROM	TO	TCP	TCP/Vegas	TEAR	Trinomial
USER	ROBOT	5352	7647	12 666	8780
USER	FPGA	6182	7642	395	8276
ROBOT	USER	5335	7641	393	8020
FPGA	USER	6182	7642	17 462	8285
DROPPED		47	0	35	1773
USER	CAMERA	1023	1033	1 027	1027
CAMERA	USER	1023	1033	1 030	1029

Table 2
RTT behaviour per flow and protocol

	TCP		TCP/Vegas		TEAR		Trinomial	
	Average (ms)	Deviation (ms)	Average (ms)	Deviation (ms)	Average (ms)	Deviation (ms)	Average (ms)	Deviation (ms)
ROBOT	66.95	10.09	51.58	3.16	25.93 / 25.68	4.25 / 4.62	58.45	8.53
FPGA	64.13	6.85	51.6	3.00	24.82 / 24.56	2.78 / 3.88	58.5	8.23
CAMERA	64.87	6.15	53.23	2.24	54.89	9.27	60.2	8.15

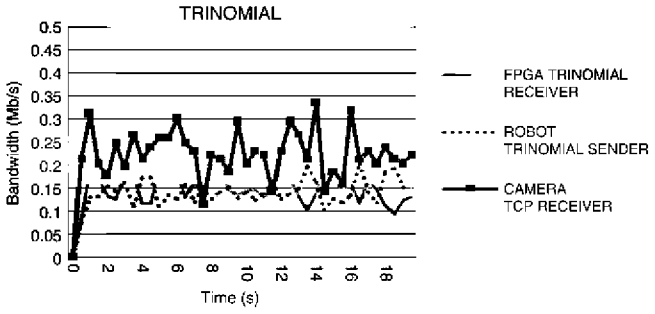

Fig. 12. Telelaboratory experiment using Trinomial for the Robot, and the FPGA. The monitoring camera uses TCP.

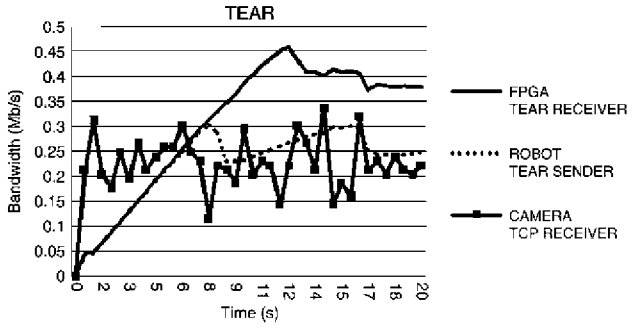

Fig. 13. Telelaboratory experiment using TEAR for the Robot, and the FPGA. The monitoring camera uses TCP.

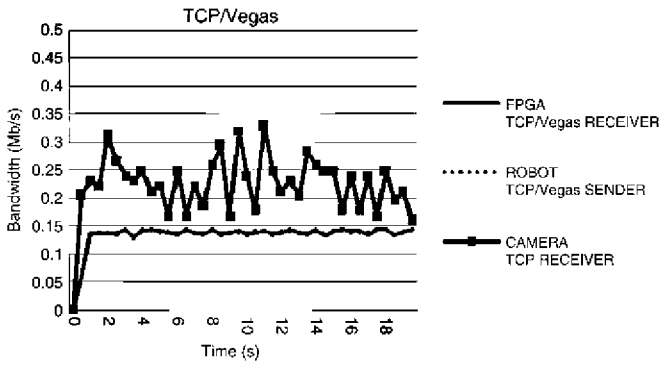

Fig. 14. Telelaboratory experiment using TCP Las Vegas for the Robot, and the FPGA. The monitoring camera uses TCP.

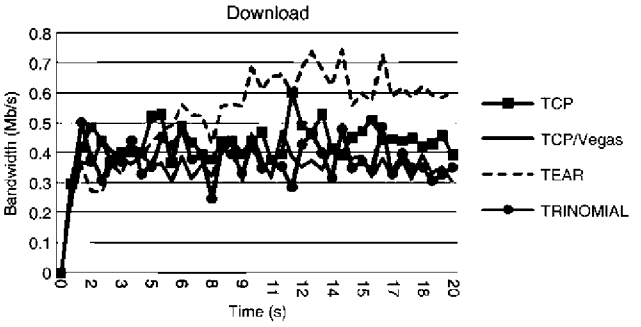

Fig. 15. Comparative analysis of TCP, TCP Las Vegas, TEAR and Trinomial protocols for the visual servoing experiment on the link from the Telelaboratory to the user (i.e. download link). Every flow represents only the packets that have information (i.e. non ACK packets are shown).

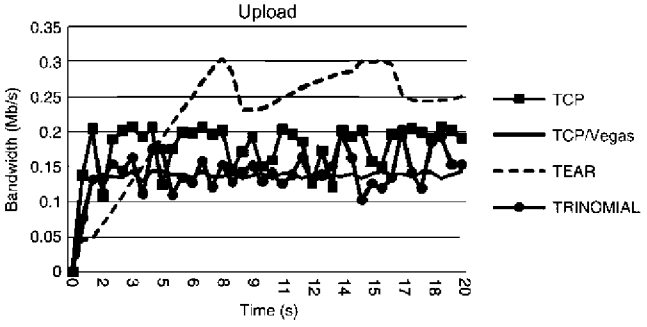

Fig. 16. Comparative analysis of TCP, TCP Las Vegas, TEAR and Trinomial protocols for the visual servoing experiment on the link from the user to the telalaboratory (i.e. upload link).

The RTT goes on an average of 51.61 ms (25.93 + 25.68). In some situations the RTT of the TCP Reno protocol is twice that of the TEAR one. Moreover, the Tear protocol has a slow start (see Fig. 13), which is not convenient for teleoperation.

For the TCP Las Vegas, the RTT deviation is the most interesting for the master/slave teleoperation. In fact, it sets the router buffers to a minimum RTT average. From the bandwidth point of view (see Fig. 14), it offers almost 2000 more packets to the robot, than the same simulation using the TCP Reno, which represents an excellent improvement. Besides this, TCP Las Vegas does not drop any packets for the whole simulation.

In summary, for this asymmetric experiment (see Figs. 15 and 16), the TEAR protocol is the one that has a more stable and shorter RTT, using less bandwidth and sends more packets between the

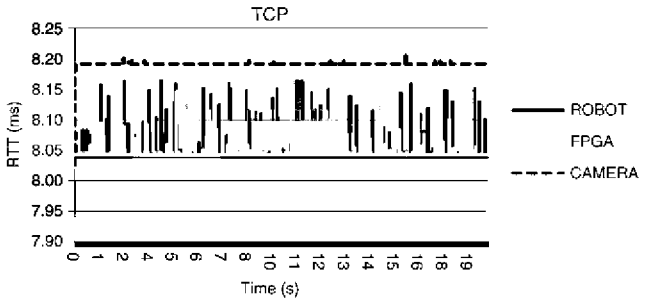

Fig. 17. Results of the RTT behaviour NS-2 simulation when Node 3 uses the TCP protocol.

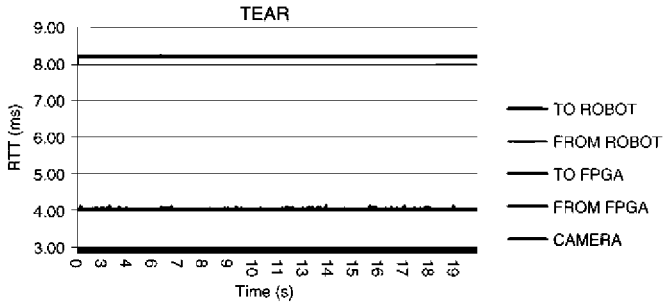

Fig. 18. Results of the RTT behaviour NS-2 simulation when Node 3 uses the TEAR protocol.

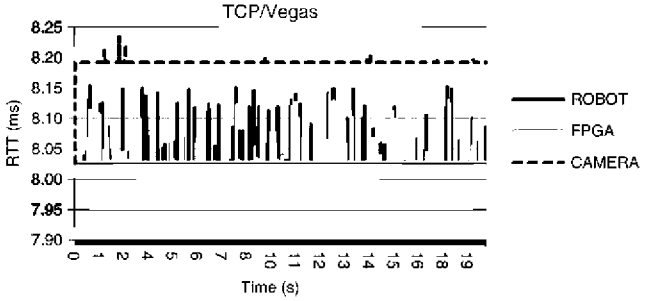

Fig. 19. Results of the RTT behaviour NS-2 simulation when Node 3 uses the TCP/Vegas protocol.

student and the robot. The trinomial has one of the biggest RTT, and loses more packets than any other. The TCP Las Vegas loses less packets than any other, presents a very stable RTT, but does not send so many packets as the TEAR protocol.

7. Results using a symmetric network (on campus)

In this section we are going to observe the RTT and bandwidth behaviour of Trinomial, TCP Reno, TCP Las Vegas, and TEAR protocols for the industrial telelaboratory, using a symmetric network. Congestion is not presented in this experiment because the available bandwidth in the network is bigger than that required by the experiment.

The requirement for the industrial manipulator is getting one packed every two milliseconds in order to fit the robot controller's requirements, for these experiments, the Trinomial and the TEAR protocols have been improved, in order to limit their sending ratio.

As seen in Fig. 6, we have the Node 3 that represents a student that is performing a teleoperation (or visual servoing) experiment on the industrial robot (i.e. node 4). In the simulation, the traffic from the node 3 will vary from Trinomial, TCP Reno, TCP Las Vegas, and TEAR.

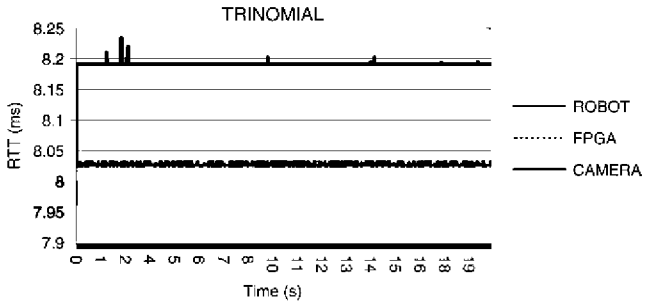

Fig. 20. Results of the RTT behaviour NS-2 simulation when Node 3 uses the Trinomial protocol.

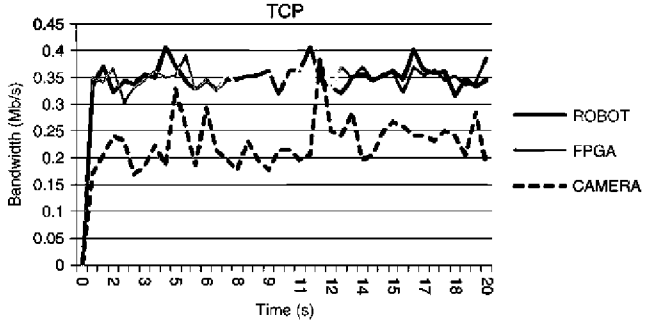

Fig. 21. Telelaboratory experiment using TCP for the Robot, the FPGA, and the camera.

For this experiment, as the RTTs are so small (see Figs. 17–20), and the robot is not able to perform a command that is less than 2 ms after its predecessor, the four protocols presented are good enough to get a smooth movement of the robot in the experiment. In fact, it has been necessary to modify the Trinomial and the TEAR protocols in order to assure the requirement that the robot will not get two packets that are closer than 2 ms. In summary, as in this experiment there is no congestion in the network, the router has an optimum performance and the RTT sets itself to its minimum.

Moreover, as we can see in the figures, the trinomial protocol presented for this experiment a better delay stability, due to the fact that this protocol has a ratio-based performance, instead of the window-based design of TCP Reno, TCP Las Vegas and TEAR. This is very good for Internet Teleoperation.

In summary, the modified version of the Trinomial protocol makes better use of the available bandwidth, because its performance is almost constant and it reaches the maximum bandwidth that the robot requires. The TCP and TCP Las Vegas work in a similar way when there is no congestion in the network, which has a certain variance of the bandwidth used, because of its window-based design. On the other hand, the TEAR protocol gets the available bandwidth in a very slow manner, which is particularly unsatisfactory in situations where there is no congestion. However, the stability of the bandwidth, once the protocol has reached the robot bandwidth requirement, is as good as the Trinomial. (see Figs. 21–24).

8. Conclusions

Within the network robotics context via the Internet, and particularly the teleoperation case, UDP and TCP protocols can be improved, in order to acquire better performance and smoothness.

The TCP Reno uses a congestion control mechanism, and a connection establishment that imply having high delay jitter (fluctuations), a situation that is not appropriate for applications such as Internet teleoperation.

Table 3
Summary of protocol Recommendations versus control robots commands

Data/ Network	TCP Reno	UDP	TCP/Vegas	Trinomial	Tear
High level commands/ Symmetric	Good (First recommended)	No good (No retransmission)	Good (Second recommended. Conservative flow)	No good (No retransmission)	No good (No retransmission)
High level commands/ Asymmetric	Good (First recommended)	No good (No retransmission)	Good (Second recommended. Conservative flow)	No good (No retransmission)	No good (No retransmission)
Low level commands/ Symmetric	No good (High jitter)	No good (No congestion control)	Good (Second recommended. Conservative flow)	Good (First recommended)	No good (Slow start and Problems with bidirectional flows)
Low level commands/ Asymmetric	No good (High jitter)	No good (No congestion control)	Good (Second recommended. Conservative flow)	No good (RTT Problems)	Good (First recommended)

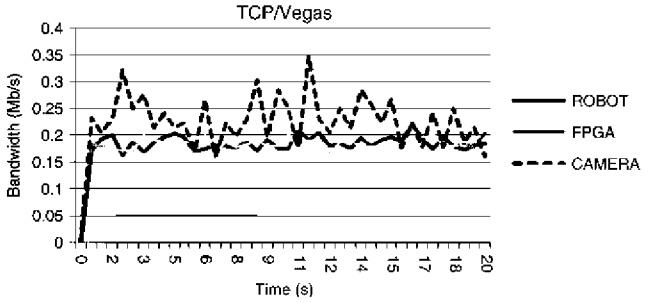

Fig. 22. Telelaboratory experiment using TCP/Vegas for the Robot, and the FPGA. The monitoring camera uses TCP.

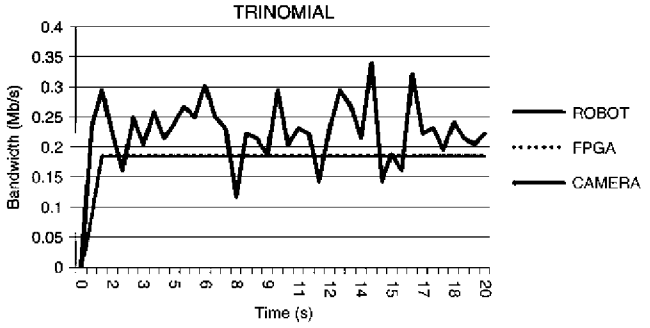

Fig. 23. Telelaboratory experiment using Trinomial for the Robot, and the FPGA. The monitoring camera uses TCP.

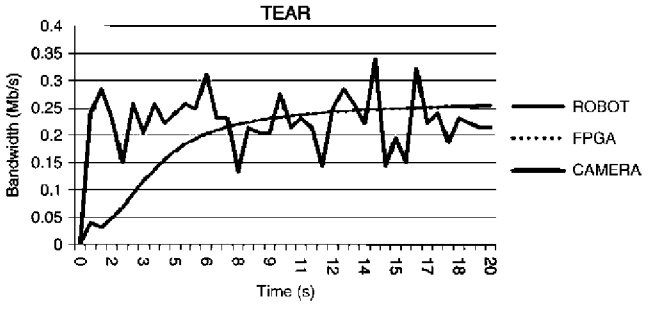

Fig. 24. Telelaboratory experiment using TEAR for the Robot, and the FPGA. The monitoring camera uses TCP.

The TCP Las Vegas improves the TCP Reno performance in congestion situations, such as the one presented for the asymmetric network. The RTT is maintained at a constant level in the presence of congestion, and works the same way as TCP Reno when there is no congestion. However, the TCP Las Vegas is very conservative when there are competing flows, which implies having an extra reduction of the send ratio.

The Trinomial protocol is a nice solution, which uses as much bandwidth as possible, providing smoothness for a bilateral teleoperation via the Internet. However, it introduces extra time-delays due to the fact that it sets the router buffers to the maximum load, and it is not designed for asymmetric networks. As well, as seen in the RTT results, depending on the parameter configurations, may not be as TCP-Friendly as other protocols. RTT behaviour is very important for some experiments, like remote visual servoing and teleoperation. Please note these conclusions about the Trinomial are extracted from the simulations implemented by the authors of this article, as they were not available via other alternatives.

The TEAR protocol is the one that sends more control information to the robot on an asymmetric configuration, in a very smooth way. However, for the Internet telerobotics context this is not sufficient, due to the fact that it needs priorities to be set for every data flow. For example, for the remote visual servoing experiment, the FPGA and robot flows must have a minimum RTT and maximum priority, and the monitoring Camera flow does not need to have such a configuration. Moreover, the TEAR protocol has a slow start, which prevents the systems from getting the available bandwidth in a fast manner.

As a summary, Table 3 presents the recommendations of the authors of this paper when having teleoperation data flows of high and low level commands through both, a symmetric and an asymmetric network.

When sending high-level commands, the recommendation is using the TCP Reno protocol, since it guarantees the single packet will reach the destination even if it needs to be retransmitted.

When having an Internet teleoperation with low-level commands within a symmetric network, as retransmission is not appropriate and a fast start is necessary; the Trinomial protocol offers the best performance.

On the other hand, for an Internet teleoperation with low-level commands within an asymmetric network, the TEAR protocol presents the best results.

As conclusion, the requirements we wish for a specific End-To-End Congestion Transport Protocol for Internet Teleoperation are the following:

1. Smooth Congestion Avoidance: It will study the smooth equilibrium between bandwidth and time delay for master/slave teleoperations. This equilibrium depends on the robot configuration and the specific application.
2. Differentiated Services: Including priorities in the flows will allow the bandwidth allocation of cameras, robot control, and sensor information in a differentiated manner.
3. RTT feedback to the application layer (i.e. control loop): As explained in Section 3, the control techniques for teleoperation under time-delays need to know the current RTT between master and slave in order to adjust some parameters as for example the "Friction" one. It is important to provide the current and next estimated RTT information to the application layer from the transport protocol at each iteration.

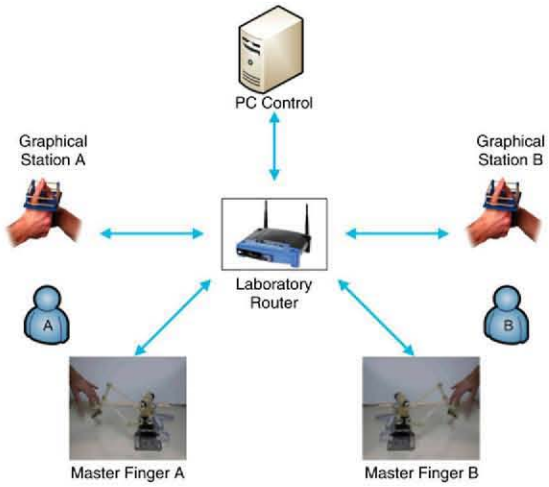

Fig. 25. The 'thumb wrestling' demonstrator scenario.

9. Future applications

New interactive applications, where some users interact continuously are being developed. The goal is that users can send and receive information in real time, according to the task they are executing. It represents an extension of the master–slave systems to a network where many devices can act as masters at the same time. The Internet Transport protocol should inform to the application layer about communication bandwidth. This information will be used by bilateral controllers in order to guarantee stability of the distributed system. Techniques based on passivity require information related to RTT or similar to avoid that communication delays make the system unstable.

An example of these applications is shown in Fig. 25. It represents a virtual 'thumb wrestling' game. It is a demonstration which focuses on transmitting haptic interactions between two users. A user attempts to capture his opponent's thumb while avoiding being wrestled out. During the match, the wrist and the rest of fingers are kept still. Each user is handling a haptic device (called MasterFinger) and a computer display:

- The haptic device registers the user movements and transmits interaction forces.
- The computer runs a graphic simulation for showing the users' movements.

All haptic information is managed by a computer server that receives, processes and sends the information back via Ethernet. The whole system is composed of five networked devices:

- PC-Control. It is the application server that managed all the information. It is running under the vxWorks real-time platform.
- 2 graphical stations. They provide the user with a graphical simulation showing the movement of both hands.
- 2 haptic controllers. They send the user movements and reflect forces according to thumb collisions.

References

R. Marin, P.J. Sanz, P. Nebot, R. Wirz, A multimodal interface to control a robot arm via the web: A case study on remote programming, IEEE Transactions on Industrial Electronics 52 (December) (2005) 1506–1520.
R. Wirz, R. Marin, P.J. Sanz, Remote programming over multiple heterogeneous robots: A case study on distributed multirobot architecture, Industrial Robot 33 (2006).
R. Marin, P.J. Sanz, A.P. Del Pobil, The UJI online robot: An education and training experience, Autonomous Robots 15 (November) (2003) 283–297.
P.X. Liu, M.Q.H. Meng, S.X. Yang, Data communications for internet robots, Autonomous Robots 15 (2003).
P.X. Liu, M.Q.H. Meng, P.R. Liu, S.X. Yang, An end-to-end transmission architecture for the remote control of robots over ip networks, IEEE Transactions on Mechatronics 10 (2005).
S. Floyd, K. Fall, Promoting the use of end-to-end congestion control in the internet, IEEE/ACM Transactions on Networking 7 (1999) 14.
I. Rhee, V. Ozdemir, Y. Yi, TEAR: TCP Emulation at receivers: Flow control for multimedia streaming, Department of Computer Science, NCSU 2000.
S.H. Low, L. Peterson, L. Wang, Understanding Vegas: A duality model, Journal of ACM 49 (2002) 18.
J. Mo, J. Walrand, Fair end-to-end window-based congestion control, IEEE/ACM Transactions on Networking 8 (2000) 11.
R.T. Fielding, R.N. Taylor, Principled design of the modern web architecture, in: ICSE 2000, 2000, pp. 407–415.
R. Marin, G. Leon, R. Wirz, J. Sales, J.M. Claver, P.J. Sanz, Remote control within the UJI robotics manufacturing cell using FPGA-based vision, in: ECC'2007 European Control Conference, 2007.
P.X. Liu, M. Meng, Y. Xiufen, J. Gu, An UDP-based protocol for internet robots, in: World Congress on Intelligent Control and Automation, 2002, pp. 59–65.
S. Hirche, M. Ferre, J. Barrio, C. Melchiorri, M. Buss, Bilateral control architectures for telerobotics, in: Advances in Telerobotics, in: STAR Series, Springer Verlag, 2007.
R. Anderson, M. Spong, Bilateral control of teleoperators with time delay, IEEE Transactions on Automatic Control 34 (1989) 13.
G. Niemeyer, J. Slotine, Stable adaptive teleoperation, IEEE Journal of Oceanic Engineering 16 (1991) 10.
P. Arcara, C. Melchiorri, Control schemes for teleoperation with time delay: A comparative study, Robotics and Autonomous Systems 38 (2002).
C. Secchi, S. Stramigioli, C. Fantuzzi, Digital passive geometric telemanipulation, in: IEEE International Conference on Robotics and Automation, Taipei (Taiwan), 2003.
S. Stramigioli, C. Secchi, A. van der Schaft, C. Fantuzzi, Sampled data systems passivity and sampled port-hamiltonian systems, IEEE Transactions of Robotics and Automation 21 (4) (2005).
Arjan van der Schaft, L2-gain and passivity techniques in nonlinear control, Communication and Control Engineering (2000).
J. Postel, RFC 768: User Datagram Protocol, 1980.
J. Postel, RFC 793: Transmisión Control Protocol, DARPA Internet Program Protocol Specification, 1981.
J. Park, O. Khatib, Robust haptic teleoperation of a mobile manipulation platform, in: Experimental Robotics IX, in: Star, Springer Tracts in Advanced Robotics, 2005.
S.E. Butner, M. Ghodoussi, A real-time system for tele-surgery, in: 21st International Conference on Distributed Computing Systems, 2001.
J. Padhye, J. Kurose, D. Towsley, R. Koodli, A model based TCPfriendly rate control protocol, in: NOOSDAV'1999, 1999, pp. 137–151.
R. Rejaie, M. Handley, D. Estrin, RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the Internet, in: IEEE Infocom, 1999, pp. 1337–1345.
D. Sisalem, H. Schulzrinne, The loss-delay adjustment algorithm: A TCP-friendly adaptation scheme, in: Int. Workshop Network and Operating System Support for Digital Audio and Video, NOSSDAV, 1998, pp. 215–226.
S. Jin, L. Guo, I. Matta, A. Bestavros, TCP-friendly SIMD congestion control and its convergence behaviour, in: 9th IEEE Int. Conf. Network Protocols, Riverside, CA, 2001, pp. 156–164.
H. Schulzrinne, S. Casner, R. Fredrick, V. Jacobson, RFC 1889: RTP: A transport protocol for real-time applications, 1996.
Y. Uchimura, T. Yakoh, Bilateral robot system on the real-time network structure, IEEE Transactions on Industrial Electronics 51 (2004).
L. Ping, L. Wenjuan, S. Zengqi, Transport layer protocol reconfiguration for network-based robot control system, in: IEEE Networking, Sensing and Control 2005, 2005.