PERFORMANCE EVALUATION OF A DISTRIBUTED TELEROBOTIC FRAMEWORK

Mayez Al-Mouhamed, Onur Toker, and Asif Iqbal

College of Computer Science and Engineering
King Fahd University of petroleum & Minerals
Dhahran 31261, Kingdom of Saudi Arabia.
mayez/onur/aiqbal@ccse.kfupm.edu.sa

ABSTRACT

In this paper we present the performance evaluation of a *Distributed Component based Telerobotic Framework* that implements a real-time interaction between a telrobotic client and server. The objective is to optimize delays in multi-streaming of force feedback, stereo data and master-slave commands. Different scenarios are considered and statistically analyzed to relate the effect of thread manipulation to time delays. Telerobotic components communicate with each other using *.NET Remoting* and *SOAP (Simple Object Access Protocol)* that automatically handle the network resources and data transfer. This approach significantly reduces the delays over a LAN as we are able to attain a rate of 17-18 stereo frames per second from camera(server) to remote client over the same LAN. To the best of our knowledge, this is the highest rate achieved over a 100 Mbps LAN.

Keywords: DCOM, Distributed Framework, Force Feedback, Multi-Threading, Performance, Telerobotics.

1. INTRODUCTION

Telerobotics uses highly demanding media data such as tactile, proprioceptive (musle tension), kinesthetic (joint angle and velocity) information, and stereo vision [1]. These have ever increasing sampling rates with a tradeoff between quality and sampling frequency.

Internet and LANs produces random transmission delays due to the lack of quality of service which causes real-time processes to go unstable when the delay exceeds a certain limit. The delays cause proportional degradation in operator performance and jitter disturbs the velocity due to time varying intervals. Real-time streaming of force feedback [2] and stereo vision are needed.

An Internet based *DCOM* design for telerobotics is proposed by [3]. Internet reliable TCP/IP sockets [4] produce delay jitter in the arrival rate of originally synchronous packets. An average delay for a small packet from 50 ms to 100 ms is common (US) leading to a 10Hz sampling rate. The real-time multitasking system allow spawning of many tasks and prioritizing them so that to control the order of task execution and the amount of CPU time allocated to each task.

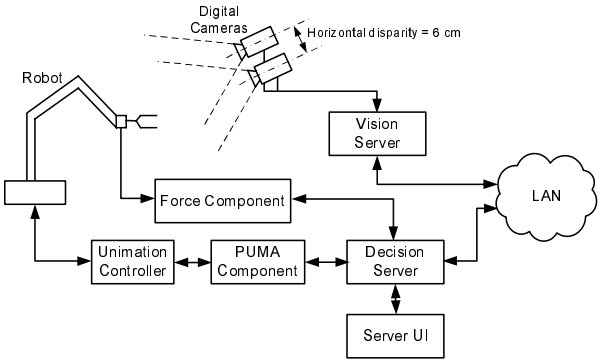

Figure 1. Server side of the distributed framework

In this paper we present the performance evaluation of a reliable real-time telerobotic system connecting a master station to a slave station using a *.NET Components based Distributed Telerobotic Framework*. We study the real-time delays experienced during data streaming from slave to master of (1) force feedback, (2) stereo vision, and ((3) commands flowing in the opposite direction.

The organization of this paper is as follows. In Section 2 we present a brief overview of our distributed framework. In Section 3 we evaluate our approach followed by the comparison in Section 4.

2. DISTRIBUTED TELROBOTIC FRAMEWORK

Distributed application programming is one of the schemes to establish a reliable connection between master and slave arms. This includes modules that are implemented as software components that communicate with each other using distributed paradigm. The framework takes care of Network protocol issues, network resources, and data transfer over the network. All of these components are created using Microsoft *.NET* Technologies using Visual C# as programming language. In the following we shortly describe client and server components.

2.1. The server components

The server components are: (1) PUMA Component, (2) Force Sensor Component, and (3) Decision Server Component.

PUMA component acts as a software proxy of the robot for which commands are issued. The PUMA component reads current robot joint $\theta_P(t)$ as a 6×1 vector. A command for an incremental joint motion $\Delta\theta$ is sent directly to robot. A command for an incremental cartesian motion is specified in hand frame translation ΔX (3×1) and orientation matrix ΔM (3×3). PUMA computes the new robot hand position $X_{new} = G(\theta_P) + \Delta X$ and orientation $M_{new} = M_P.\Delta M$, where $G()$ is the direct kinematic model of slave arm and M_P is the current robot hand orientation matrix. PUMA computes the inverse kinematics for X_{new} and M_{new} and finds the corresponding joint vector $\Delta\theta$ which is sent to robot.

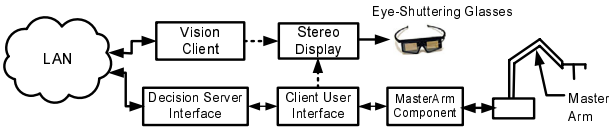

Figure 2. Client side of the distributed framework

The force sensing component (FSC), implemented in a separate thread, reads the robot wrist force sensor and create a stream of reflected force feedback directed to the master station.

The DecisionServer is a component that provides an autonomous loop on the server to support supervisory telerobotic control.

2.2. Client components

The client contains the *IDecisionServer* interface to reference the server side component through *.NET Remoting*.

The Decision Server interface contains all the definitions to execute public methods on PUMA and FSC. It allows the client side to access the server side instance of DecisionServer as a local component through *IDecisionServer* interface.

.NET Remoting provides reliable binding between client and server over a LAN. The multi-threaded distributed telerobotic system (Fig. 1 and 2) allows simultaneous activation of many threads like grabbing of stereo video data, reading force sensors, sending and receiving robot control signals over the LAN to one or more clients.

Two digital cameras generate stereo video data. Both the stereo data and the distributed components share the same LAN connection using different ports for data transfer. The client uses the GUI as well as a 6 dof master arm to issue commands to the slave arm on remote site. The vision client receives the synchronized stereo data from the LAN through windows sockets.

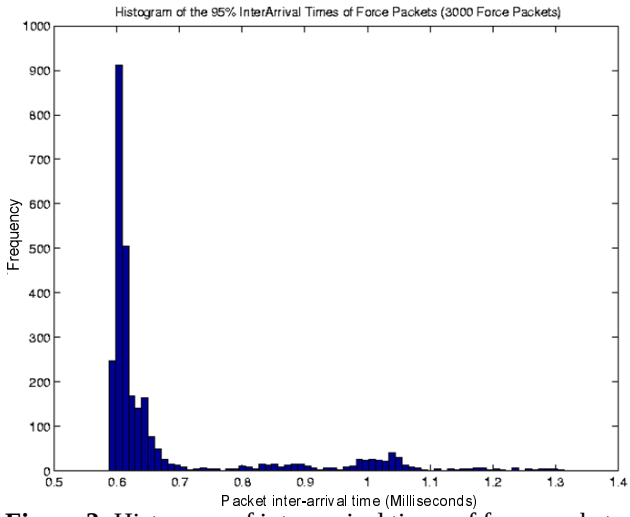

Figure 3. Histogram of inter-arrival times of force packets

3. PERFORMANCE EVALUATION

Performance evaluation experiments under different conditions were carried out on the distributed framework described in section 2. The bandwidth of the LAN is 100 Mbps and both the client and server PCs are 2.0 GHZ P-IV machines with 1 GB DRAM. Each force data packet contains 6 double values which equal $6 \times 8 = 48$ bytes. The experiments are explained in the following sections.

3.1. Force Only

In this setup, only force information is transferred from the server to client. There is no video transfer neither any command signal present during the experiment. A histogram of inter-arrival times of force packets is shown in Figure 3. This data fits to an Inverse Gaussian distribution with a mean value of 0.679 ms and 90% of the data lying between 0.59 to 0.92 ms.

3.2. Force and Video

In this case force thread alongside video thread is running on the server. A histogram of the the inter-arrival times of force packets in the presence of video transfer is shown in Figure 4. This is an Inverse Gaussian distribution with a mean value of 1.08 ms and 90% of the data lying between 0.5 and 3.9 ms. Clearly the presence of the video has pushed the mean value from 0.68 to 1.08 ms.

A magnified plot of the inter-arrival times of force packets in presence of video thread is shown in Figure 5. The pulse below the actual plot shows the interval during which the transfer of a stereo video frame was in progress. On the x-axis is the force packet number while on y-axis we have milliseconds.

A histogram of the inter-arrival times of only those packets that were received during the transfer of a stereo video frame is shown in Figure 6. The data best fits to a Logistic distribution with a mean value of 5.41 ms and 90% confidence interval lying between 0.5 and 13.0 ms.

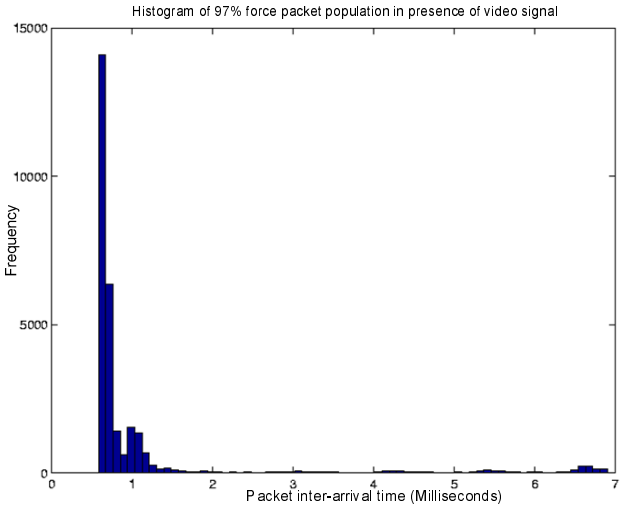

Figure 4. Histogram of inter-arrival times of force packets with video

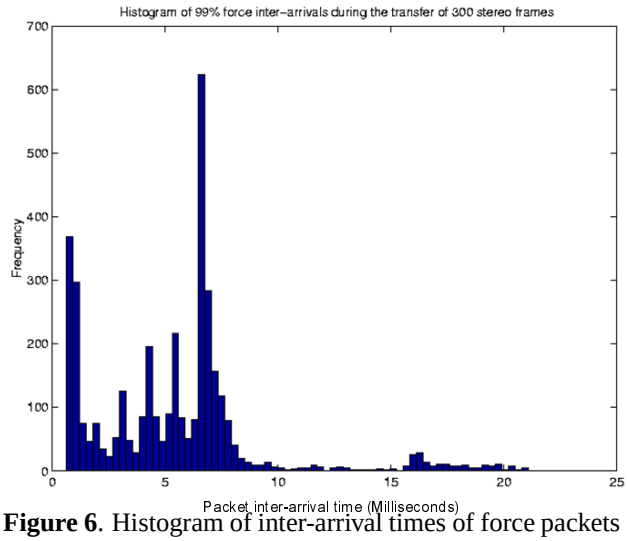

Figure 6. Histogram of inter-arrival times of force packets during the transfer of a video frame

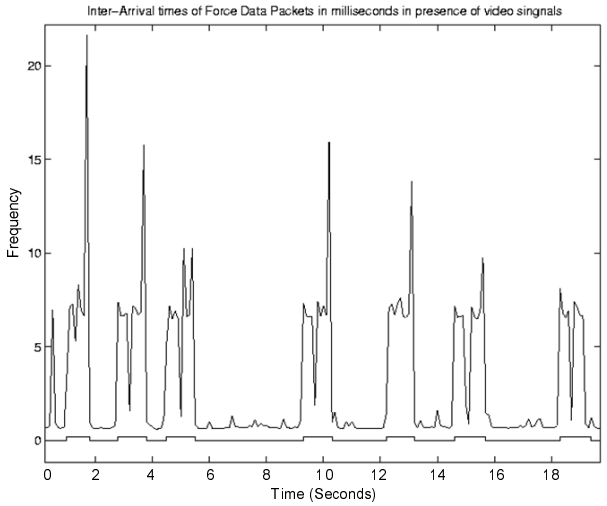

Figure 5. A Magnified plot of inter-arrival times of force packets with video

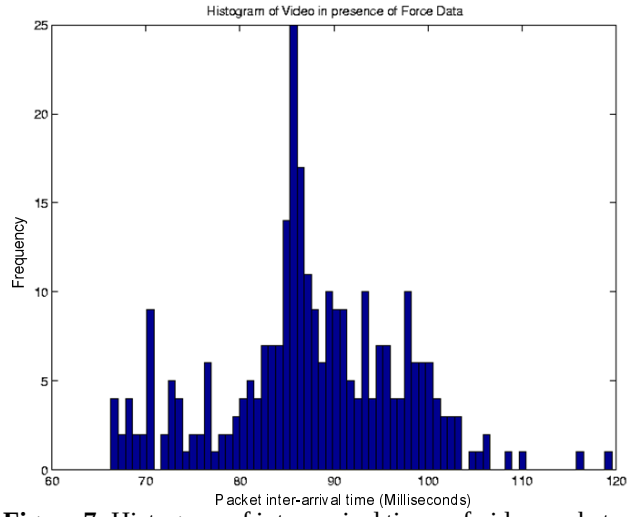

Figure 7. Histogram of inter-arrival times of video packets in the presence of force thread

Clearly we can see a large difference between the inter-arrival times of force packets without video which is 0.679 ms and here the packets during the transfer of a stereo video frame have a mean inter-arrival time of 5.41 ms. The shows the loading of network with the transfer of large video data.

The mean value of the inter-arrival times of stereo video frames is 87.57 ms with a 90% confidence interval falling between 72 and 107 ms. A histogram of the data is shown in Figure 7.

3.3. Force, Command and Video

When all of the three force, command and video threads are invoked simultaneously, for the force packets we get a mean inter-arrival rate of 1.1 ms while 100% of the population remains under 8 ms. A magnified plot of the data against the force packet arrivals is given in Figure 8. Clearly the peaks in the plot show the effect of the transfer of video frames on the inter-arrival times of force packets.

4. COMPARISON

A client-server framework using VB 6.0 and TCP ActiveX controls, implemented by Al-Harthy[5], takes 55 ms for a command signal (48 bytes) to reach from client to server. In our case a force packet consisting of 6 double values (6×8 bytes = 48 bytes, same size) took about 0.7 ms in the absence of stereo video data and 1.1 ms in the presence of video stream. This difference is achieved by using the distributed component based approach in place of TCP based custom protocols.

The video transfer rate achieved by Teresa[6] is 1 frame every 3 seconds for a single image of 16 bit color depth over the internet. The Java-based frame grabbing software takes one second for an image to move from camera to DRAM as compared to a mean value of 24 ms obtained by

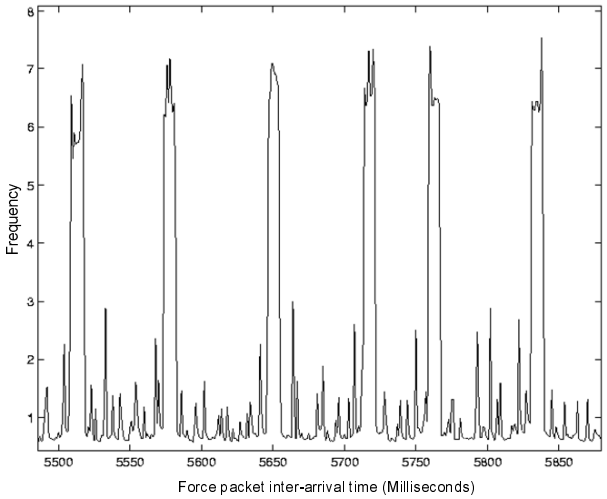

Figure 8. Magnified plot of inter-arrival times of force packets in the presence of video and command threads

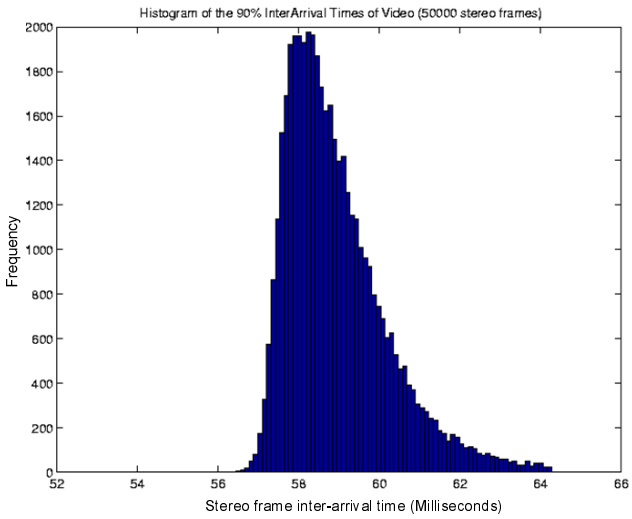

Figure 9. Optimized transfer of stereo video

our approach using DirectShow.

In a LAN setup, Huosheng et. al. [7] quote a transfer rate of 9-12 fps with time delays less than 200 ms for a single image of size 200×150 pixels. This is to be noted that the images are not bitmap but are compressed using JPEG compression technique. In comparison to this, our stereo video client-server transfers *two uncompressed* images *(stereo frame)* of size 288×360 pixels at a rate of 11.74 fps with a delay of around 87 ms only.

It is also worth noting that if the serialization of capturing and transferring-over-LAN operations is removed by thread manipulation on the server, an inter-arrival delay of around 55 ms can be achieved while utilizing nearly 90% of the bandwidth of a 100 Mbps LAN. After some experiments in this direction, we are able to obtain a mean inter-arrival time of 58.57 ms. A histogram of inter-arrival times of stereo video frames is shown in Figure 9.

5. ACKNOWLEDGEMENT

This work is supported by King Abdulaziz City for Science and Technology (KACST) under research project grant AR-20-80. We also acknowledge computing support from KFUPM.

6. CONCLUSION

A distributed component based telerobotic framework is evaluated under various conditions for performance. The framework is developed using most advanced software authoring tools available for component development. Very significant reduction in network delays is observed. Over a 100Mbps LAN, our approach can transfer two images (stereo frame) of size 288×360 pixels at a rate of 17-18 fps with a delay of around 58 ms only. Thanks to multithreading for the graceful degradation of real-time signals as a force-stream packet of 48 bytes takes about 0.7 ms (no video) and *only* 1.1 ms in the presence of video stream.

7. REFERENCES

[1] F. Paolucci; M. Andrenucci. Teleoperation using computer networks: Prototype realization and performance analysis. *Electrotechnical Conference, MELECON '96., 8th Mediterranean*, 2:1156–1159, 1996.

[2] S. E. Salcudean; N. M. Wong; R. L. Hollis. Design and control of a force-reflecting teleoperation system with magnetically levitated master and wrist. *IEEE Transactions on Robotics and Automation*, 11, No. 6:844–858, December 1995.

[3] Y. E. Ho; H. Masuda; H. Oda; L. W. Stark. Distributed control for tele-operations. *IEEE/ASME Transactions On Mechatronics*, 5(2):100–109, June 2000.

[4] W. J. Book; H. Lane; L. J. Love; D. P. Magee; K. Obergfell. A novel teleoperated long-reach manipulator testbed and its remote capabilities via the internet. *Proc. of the IEEE Inter. Conf. on Robotics and Automation*, 3, No. 6:1036–1041, 1997.

[5] A. Al-Harthy. Design of a telerobotic system over a local area network. *M.Sc. Thesis, King Fahd University of Petroeum and Minerals*, January 2002.

[6] T. Ho. System architecture for internet-based teleoperation systems using java. Master's thesis, Department of Computing Science, University of Alberta, Canada, 1999.

[7] H. Hu; L. Yu; P. W. Tsui; Q. Zhou. Internet-based robotic systems for teleoperation. *International Journal of Assembly Automation*, 21(2), 2001.