Communication issues in distributed multisensory robotics

Eric Colon, Royal Military School, (mecatron.rma.ac.be)
Alessandro Muzzetta, Massimo Cristaldi, IES Solutions (www.i4es.it)

Abstract

The EU funded project ViewFinder aims at integrating mobile robots into a command and control network. These robots will be equipped with numerous perception sensors producing a lot of data that would have to be exchanged between different computers on the robots or over a wireless link. This paper presents the communication requirements of the ViewFinder project and reviews possible technical solutions. It describes and compares two software packages, namely CoRoBA and Mailman, that are used in the project. Results of communication performance of standard Wi-Fi and Wimax systems in typical ViewFinder environments are reported. Finally tailored solutions for solving communication issues are proposed.

Keywords: Middleware, CORBA, wireless communication, Quality of Service

The ViewFinder project

The ViewFinder project aims at integrating mobile robots into a command and control network. These robots will be equipped with numerous sensors that will be used for robots navigation and for the detection of dangerous elements like chemical agents or toxic fumes. These data could be partially processed on board of the robots and will also be processed, stored and visualized in a remote base station. This paper analyses the communication requirements between software components and between the robots and the remote base station as well as possible tools for implementing distributed multisensory robotic systems. It describes and compares two software packages, namely CoRoBA and Mailman, and proposes different solutions to solve the communication issues.

CoRoBA [1] is a framework for controlling robots through a standardized set of components, a well defined program model and design patterns. It facilitates the development of distributed robotic applications. CoRoBA relies on an event driven architecture and promotes a systematic implementation of components. It focuses on how to program components in a distributed robotics system and is not concerned with networking details, delegating this job to the CORBA implementation.

Mailman is a high performace message bus built for wireless networks. It comprises a network level transport protocol specification and an implementation of such a protocol. It is built on top of UDP/IP and specifically designed to make efficient use of wireless communication devices. Mailman has been concieved because of the nature of wireless links and to optimize the transport protocol for such networks. The main features of Mailman are its facilities for controlling datagram priorities, allocating fair shares of bandwidth to all users, controlling congestion and sending notifications to clients, allowing user programs to easily address each other, providing automatic discovery of servers and support for reliable and non reliable datagram delivery.

Obviously CORBA and Mailman are targeting different issues but are complementary. Possible combinations of CoRoBA, CORBA and Mailman configurations are studied and compared, in order to determine the optimal solution for the project.

The communication issues in View-Finder

The communication architecture in the View-Finder project is summarized in figure 1. ViewFinder systems will have a Base Station. It includes the control station and the sensor station as well as a user interface. The ViewFinder central operational control (COC) is a multi-user system that is assumed to provide a client/server architecture.

On the robots we can distinguish three kinds of sensors. Data produced by the first group of sensors (A) are directly sent to the base station. Data produced by sensors of the group B are used by some processing components on the robot itself and the raw data and/or the processed data are forwarded to the base station. The sensors of group C are only useful for local robot control robot.

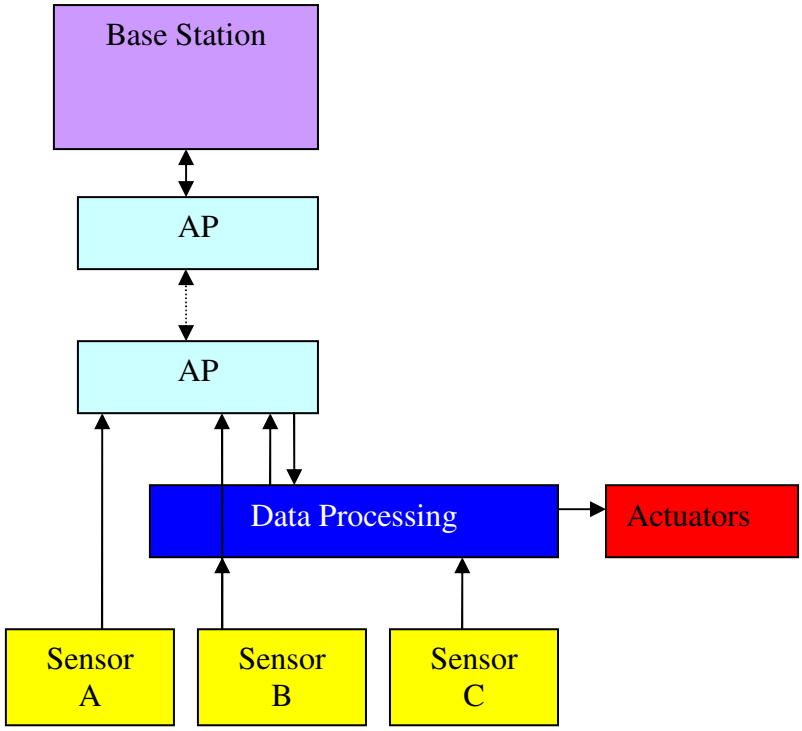

Figure 1. View-Finder Communication architecture

A careful evaluation of data produced by sensors and, consequently, the required bandwidth has been performed. Sensors of type A are a video camera, an IR camera, a 3D laser scanning system, chemical sensors and a temperature sensor. Sensors of type B produce position information (GPS, odometry, INS) and video images. Sensors of type C are obstacle detection sensors (stereo-vision system and ultra-sonic sensors). Data are sent over the wireless link (Access Point – AP) to the Base Station where they are further processed. Results are stored in a database and displayed on the screens of the control stations. Furthermore, robot control

commands and a map containing combined information are also sent to the robots from the Base Station.

One key issue when remotely controlling a robot is the latency in the control command loop. This can be solved by adding intelligence on the robot. In View-Finder, some robots will be able to navigate autonomously to follow a planned trajectory and to avoid obstacles. For interactive robot control (teleoperation), 100 msec latency is acceptable; between most other human interfaces at the base station and robot, 1 sec is acceptable. In the integrated base station different levels/schemes for the delegation of autonomy to the robots will be available. The other key issue is the bandwidth that will be required for sending sensor data from the robot to the base station. The management of the bandwidth is not a trivial task because of the wireless connection.

Another important aspect is the interconnection between different components and the design of a reliable and efficient software architecture. We have to cope with a distributed architecture where many processes need to exchange data and synchronize their activities. This is not straightforward but there exist middlewares for developing such applications.

Modern distributed robotics and sensor systems require a networking layer that can provide resource reservation and control mechanisms. In systems that provide resource reservation guarantees, the following parameters must be respected:

- Worst overall delay before a packet is delivered.
- Maximum number of packets that are lost in a given period of time.
- Control of the variation of delay (jitter).
- Minimum throughput for a given time period.

These parameters can be collectively referred to as Quality of Service (QoS).

An additional set of challenges are added when these guarantees must be provided in wireless or mobile networks. As a result of these complications, resource reservation schemes designed for wired networks are not always feasible or desirable in wireless networks.

In wired networks loss is caused by excessive congestion and, to a much lesser extent, by corruption on a wire. In a wireless network, however, data suffers much more loss due to corruption in the air-interface. One of the main causes of this corruption is transmission fading of the radio signals. Other factors include attenuation due to distance, co-channel interference, electrical noise and doppler shifting.

In distributed robotics, the remote units are typically mobile. A unit may move behind an obstacle, move too high or too low with respect to the base station antenna angle, or it may move too far from the base station. The signal can range from optimal to completely absent, depending on these and other factors. A wireless QoS system must therefore be resilient to severe loss and to total signal outages. In the View-Finder, in addition to the above QoS criteria, the following are also important:

- Fair bandwidth allocation.
- Message prioritization.

The types of data that compose the traffic in the View-Finder network comprise multimedia (video) streams, sensor readings, control commands, status messages, navigation commands, and critical messages. Each of these data types has different characteristics and QoS requirements.

Critical messages are infrequent and have the highest priority. For example, a "SWITCH-OFF" message could be sent to a robot whose chemical sensors have detected flammable gases, in order to avoid igniting an explosion. Critical messages should be delivered with top priority and resent if lost. Navigation commands and control commands have higher priority than all other traffic except critical messages. These messages must be delivered without loss and faster than other traffic.

Multimedia streams and most sensor readings have the least priority. They can tolerate loss. However, the variation in delay between multimedia frames should be kept to a minimum. Frames that are not delivered within a reasonable delay (because of congestion) should be discarded from the transmission queue.

The CoRoBA framework

The framework CoRoBA proposes an answer to the recurrent requirements that have been identified for implementing distributed multisensory robotic systems. Two different approaches have been considered when identifying requirements for the framework. The first approach takes into account the functionality of typical applications that would be built with the framework whereas the second one considers the needs of potential users. This analysis produced the following requirements list:

- Integration of different robotic systems,
- Concurrent control of several robots,
- Shared control between several users,
- Easy integration of user's algorithms.
- Flexibility (Distribution, Modularity, Configurability, Portability, Scalability, Maintainability)
- Performance and efficiency

It is obvious that some requirements conflict with each other: performance and efficiency for instance have to be traded with flexibility. However, as we do not target hard real-time applications, we can accept some performance degradations due to extra-communication overheads.

The implementation of the framework is based on several Design Patterns [2] that make the design flexible, elegant and ultimately reusable. The elementary brick in CoRoBA is a component (Component-based Architecture Pattern) that exchanged data over a network by invoking remote operations (Remote Method Call Pattern). Components are independent execution units and have separated interfaces for the configuration and the actual functionality they provide (Hierarchical Control Pattern). Components are loosely coupled, they are only connected by the structure of the data they exchanged (Data Bus Pattern), and can be discovered at run-time (Broker Pattern). According to the classical control theory, components are divided in three categories, Sensors, Processors and Actuators. They form a chain along which information is transferred via Notification Channels (EC) and like in classic control schemes, the data flow is unidirectional (Channel Architecture Pattern). Sensors read data from external devices and transmit them to other components. Processors

process received data and forward results to other Processor components or to components linked to output devices, which are called Actuators. This division provides a clear view of the functionality of each component and consequently facilitates their reuse in new applications. The communication scheme is resumed in figure 2.

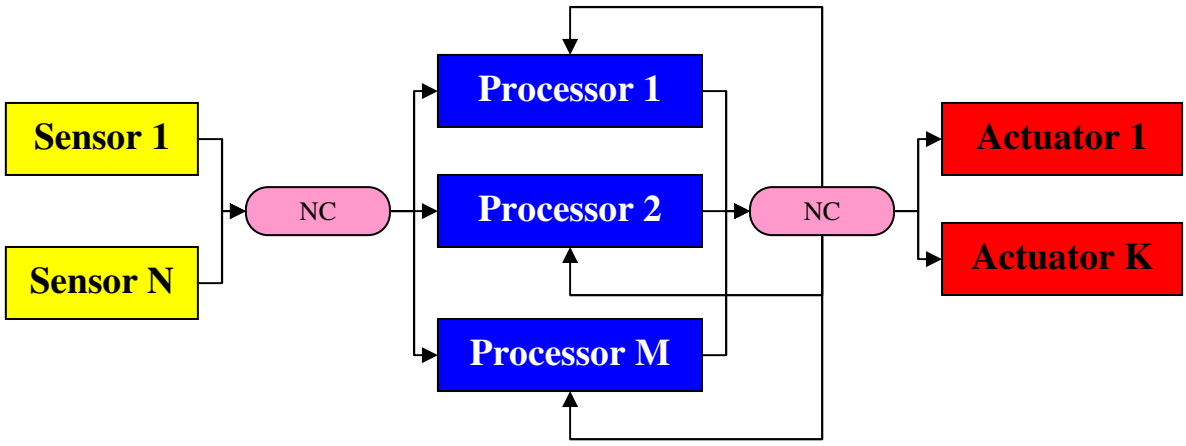

Figure 2. Communication between CoRoBA components

The first decision when developing distributed application concerns the choice of the communication library. Some framework developers have opted for the low-level socket library. While this is a good choice with regard to performance, it is a bad one concerning portability and maintenance. One solution is to use a multi-platform wrapping communication library like ACE. It is a very powerful library that can eliminate some of the drawbacks listed above but that leaves much work to the programmer.

In this framework, communication between components relies on the industry standard CORBA. CORBA has been selected because of its language and platform independence. Using such a standard simplifies the development and improves the interoperability with existing software. CORBA is actually a specification of the Object Management Group and the TAO (The ACE ORB) implementation has been chosen among others because it is an open source, efficient and standards-compliant real-time implementation of CORBA. The framework offers two different communication mechanisms; the first one is based on classical synchronous communication while the second relies on Events. In this mode, components exchange data by pushing Events through Event Channels that can be seen as pipes connecting suppliers and consumers of Events. Event based communication increases the flexibility of an application by decreasing the coupling between components.

Components are multi-threaded and implement the Method Template Pattern and the Message Queuing Pattern. They possess different running modes for the transmission of events, namely PERIODIC, SYNCHRO and TRIGGER. In the PERIODIC mode, components produce events at regular time intervals. In the SYNCHRO mode, new output events are produced by the component when an input event is received and in the TRIGGER mode, an external signal must be received in order to process or produce an event. The availability of different modes increases the flexibility of the framework, each mode being actually useful in different contexts.

A qualitative and quantitative evaluation of the framework [14] has demonstrated that the proposed solution is efficient, usable and stable. Existing applications can be largely

improved by using CoRoBA. A comparison between CoRoBA and other frameworks like MCA, DCA, GeNoM,... shows that CoRoBA combines the advantages and capabilities of most of them while avoiding their drawbacks. The framework also reduces the development time in comparison with raw CORBA programming. The availability of an automatic project generator reduces the workload of the developer. It has been widely verified that the components work as they should and that particularly, order of events are preserved, all sent events are received and that operations are executed in the right order.

Mailman

Mailman is a high performance message bus that is optimized for managed mobile networks. It provides QoS guarantees that are customized to the requirements of View-Finder components. It is resilient to network faults, allows point-to-point messaging, group communication and class based packet scheduling.
Mailman servers act as routers interconnecting wired subnets through a managed wireless interface. The servers control the link by enforcing resource allocation policies set by the users.

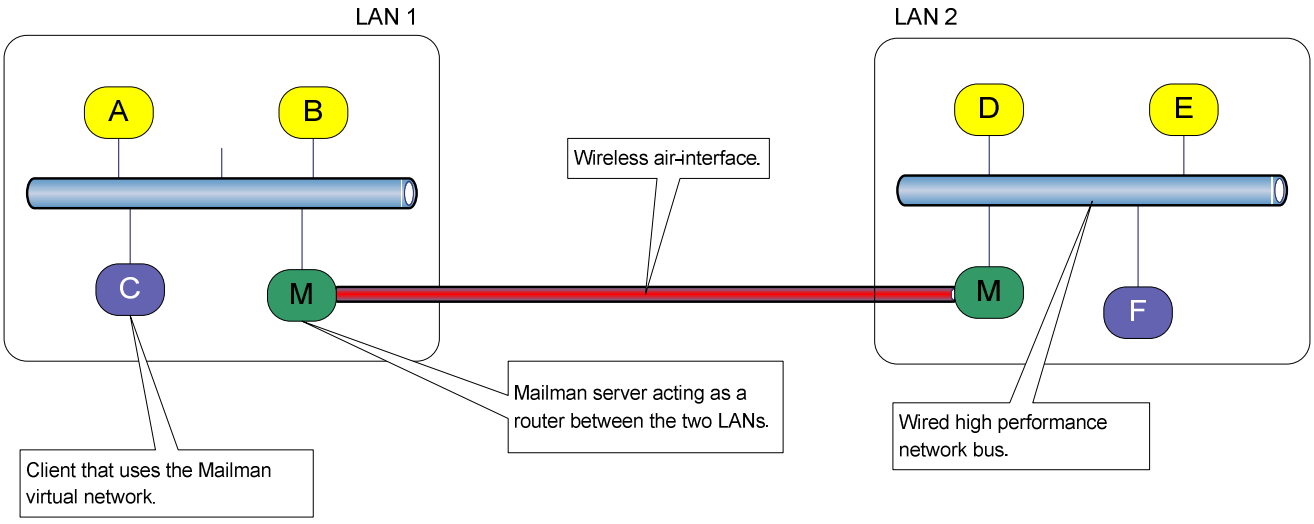

Figure 3. A typical deployment of Mailman.

Figure 3. A typical deployment of Mailman. shows a typical setup of Mailman in a network. Two wired LANs, each having a high performance wired bus, are interconnected through a wireless link. Data being sent from a component on one network to a component in the other network passes by the two Mailman servers, that act as routers. Messages sent from client C are delivered to client F, while messages originating from client D are delivered to client A, B and E.
Clients communicate explicitly with the server in their LAN and request messages to be delivered to one or more other clients. A client/sever protocol specification defines these interactions. The protocol allows the client to set the priority (or service class) of the message, to request reliable delivery, to specify the time to live of the message and the destination Id (peer or group).
The priority field of the message is set by the user, so it is the responsibility of the users to collectively agree on what traffic should be assigned to what service class. This means that Mailman only provides a mechanism to enforce a policy, not the policy itself.

Implementation

Mailman consists of a client to server protocol implemented over sockets, a message routing daemon and a server to server protocol.
The client to server protocol uses UDP/IP and adds a 32 byte header containing a priority field, an acknowledgement request bit, a time-to-live value, a destination number, a message number and the length of the payload.
The choice of designing the protocol at the application level implies that the implementation is not dependent on any specific platform. This affords View-Finder much flexibility by permitting the use of any type of wireless network hardware and software platforms. It also means that the project is not dependent on QoS facilities that are implemented at the data link layer [3, 4, 5], and therefore dependent on a specific technology. Moreover, while most QoS implementations favour real time traffic such as VoIP and video over other traffic, in View-Finder, such traffic has the lowest priority. In addition, changing user requirements are more easily met when the policies are enforceable at the application level rather than the data link level.

Services provided

The main benefits derived from the use of Mailman are:

- Class based delivery of messages.
- Fair allocation of bandwidth to all users.
- Congestion control and notification.
- Control of jitter for real time streams.
- Reliable delivery of messages over UDP.

These services are provided by the Mailman server through a packet scheduler.
Class based delivery guarantees that high priority traffic is delivered first. Fair allocation guarantees that bandwidth hungry applications (such as video streams) and rogue applications do not saturate the available spectrum, at the expense of other applications.
Mailman samples data loss over a wireless link at a given interval of time. It measures the maximum throughput over this time interval. This information is used to predict the state of the air-interface for the next time slice. With this information, it is able to send clients notifications of congestion so that they can slow down their transmission rates when bandwidth decreases. An example of such a scenario is a video streaming server that increases or decreases the frame rate and/or compression level based on the availability of bandwidth.

Use of UDP

Mailman uses UDP over IP instead of TCP because of the known shortcomings of the latter, concerning the delivery of message based and streaming data [6, 7]. In fact, the majority of network traffic in View-Finder is comprised of sensor readings and multimedia streams. Both types of traffic can tolerate loss but do not tolerate excessive delays
These types of information are useful within a limited period of time. Unless they are delivered within fixed time constraints, they become useless. When the packet scheduler is unable, due to congestion or signal outage, to deliver such data on time, it should discard them and prune the transmission queue.

Reliable delivery over UDP

The UDP [8] is a transport protocol that makes no provisions for reliable delivery. For the majority of network traffic in View-Finder, such as multimedia frames and sensor readings, the behaviour of the UDP is well suited.
Other types of messages, such as critical commands, require that the message be delivered in a reliable manner or that a connection error be returned. These messages constitute a minority of the traffic but they must be handled properly.
One way to handle the problem is to have the clients request an acknowledgment from their peers and have them resend the message if the acknowledgement is not received within a given time frame. This approach has the downside that every client that may need such a service has to implement it internally. In order to avoid the duplication of code and the resulting bugs, Mailman provides this facility to clients.
Clients can request that a message be delivered reliably. The Mailman servers will handle the acknowledgments, timeouts and resends. Only when delivery is impossible, due to a connection outage, the client will receive an error message. This service entails a certain overhead and should be used only when it is required.

Group communication

In the Mailman protocol, clients address each other though and Id. Two or more clients subscribing to an Id constitute a group. Messages sent to a group are delivered to every member of a group except for the sender.
To facilitate the use of Id's, the Mailman server provides a name-to-Id lookup service. Clients can register themselves with a symbolic name and the server assigns them a numeric Id for the duration of that session. Clients can then lookup the Id for that symbolic name. This affords flexibility for adding more groups and for reusing Id's without requiring to restart the daemons or to recompile client programs.

Server discovery

A client wishing to use Mailman services must first find out on what IP address and UDP port number the server is listening on for requests from clients. Similarly, a client cannot receive any data from peers unless it notifies the Mailman server on its LAN about its willingness to receive such data. It is possible to provide the host and port of a Mailman server to a client during configuration. However, to avoid having to reconfigure each client - which can happen when a Mailman server is moved to another host or the port number is changed - it is preferable to have the client find the server at run-time. The Mailman user library implements an algorithm for automatically discovering the nearest Mailman server on the local network.

Future development

A prototype implementation of the Mailman protocol, including a client/server library and a server daemon are available for the Linux operating system. On the client side, future plans include implementing the library in languages other than C and C++ and porting the library to major development environments on the Microsoft Windows platform. It would also be interesting to study how the queuing disciplines and class based packet scheduling can be made extensible by the user. One possibility is to implement a mini programming language that the server understands, for specifying QoS behaviour and queue handling.

Measuring mixed-communication by RMA

As a Middleware is generally quite complex and brings some overheads, one could ask the question if it is really usable for implementing robot control applications. The network performance can be expressed by the following relation:

Message Transmission Time = latency + length / data transfer rate

On one hand, the operations added by CORBA increase the latency (that is independent of the message length). On the other hand, the extra information contained in a CORBA frame is quite constant (a few hundreds of bytes) and has therefore a larger influence for small data packets. We typically have a 20% to 30% overhead in comparison with socket communication. This result is confirmed by a comparative performance experiment reported by D. Gill in [9**9. Gill, C. & Smart, W. (2002). Middleware for Robots?, In Intelligent Distributed and Embedded Systems, Papers from the 2002 AAAI Spring Symposium, Gaurav S. Sukhatme and Tucker Balch (Ed.), pages 1-5, 2002.**]. However, with increasing computing power and communication bandwidth, the overhead introduced by CORBA becomes every day less and less significant. J. Gowdy [10] qualitatively compares Inter-process Communications Toolkits for Robotics and concludes that: "If the project is a long-term project ..., then a more flexible and standard ... communication infrastructure such as CORBA may be called for ...".

Experiments have been made to measure the bandwidth required by typical CoRoBA components for controlling a Robot and to evaluate the practical control distance inside a building. The setup used for the measurements are

Hardware:

- Wi-Fi Access point (802.11 g) mounted on the Robudem.
- 3 laptops equipped with embedded Wi-Fi 802.11 g that communicate with the access point in infrastructure mode.
- 1 laptop that is connected to the Access point by an Ethernet link (100 Mbs).
- Robot control PC connected to the Access point by an Ethernet link (100 Mbs).

Control Software:

- The CoRoBA framework is used to remotely control the Robot.
- There is one sensor component reading the joystick on laptop 1.
- A Processing component on laptop 2 receives the motion commands, sends them to the robot and returns the encoders values.
- An actuator component on the laptop 1 receives and displays the encoders' values.

Video flow

A dual video data-flow is sent over the wireless communication.

- Firewire cameras are connected to the laptop 3 and 4.
- The data flow 1 goes from laptop 3 to laptop 4 and the data flow 2 goes from laptop 4 to laptop 3.
- The VLC program is used for sending and visualizing video data's.

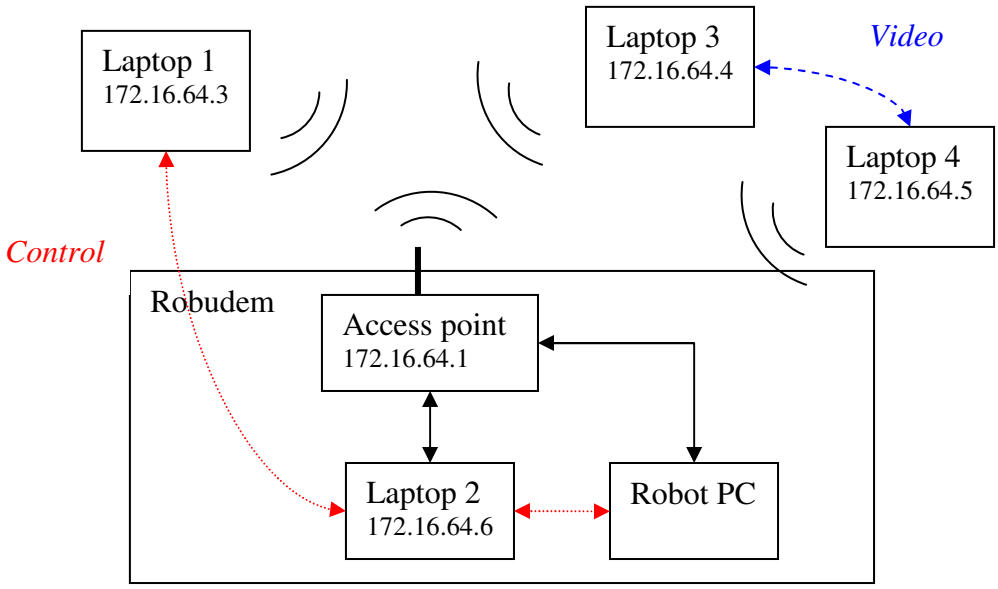

Figure 4. Test architecture

The data flow has been measured with the Ethereal application.
We can distinguish three main types of data being transmitted:

- The video streams using UDP
- The CoRoBA methods calls using GIOP
- TCP data sent by the TCP/IP stack

Results

The following picture shows typical experimental results. The data flow is indicated in Bytes/seconds on the graphic on the right. The laptop 1 controlling the robot is progressively moved away from the access point.

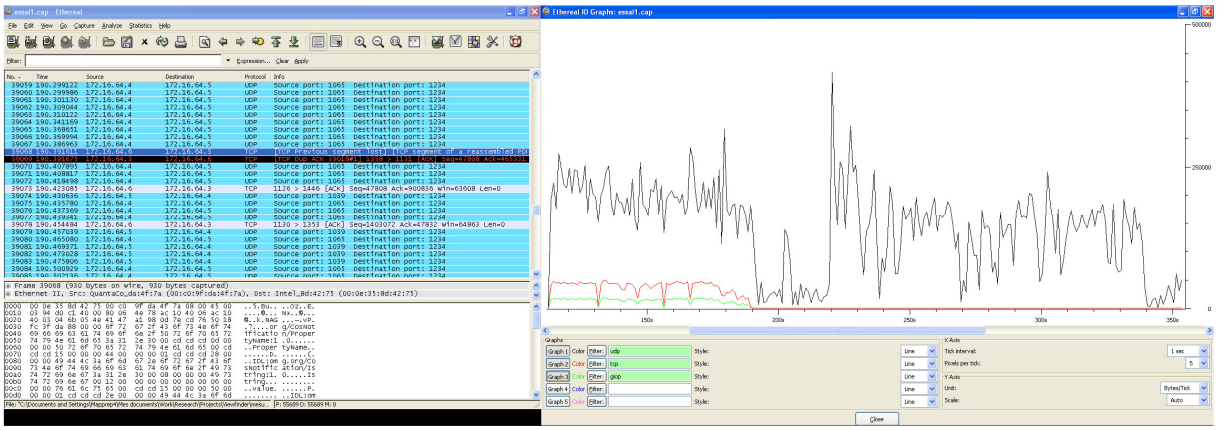

Figure 5. Communication Bandwidth utilization

From the previous graph we can see that:

- The required bandwidth for all the data flows is approximately 2.5 Mbs
- The UDP is the major data flow (2.5 Mbs).
- The TCP and GIOP flows are interrupted after 190 seconds that corresponds to a distance of approximately 30 meters. At this point, there were 5 walls between the laptop 1 and the access point.
- This interruption has been produced by the loss of the Wi-Fi signal.

- The UDP flow was also lost but was recovered by going back closer to the robot.
- The TCP link was never recovered.

The following graph shows that the TCP and UDP data flows are mixed and consequently some delays arise in the command and control sequence of the robot (GIOP and TCP packets between the address 172.16.64.3 -laptop 1- and 172.16.64.6 -laptop 2-)

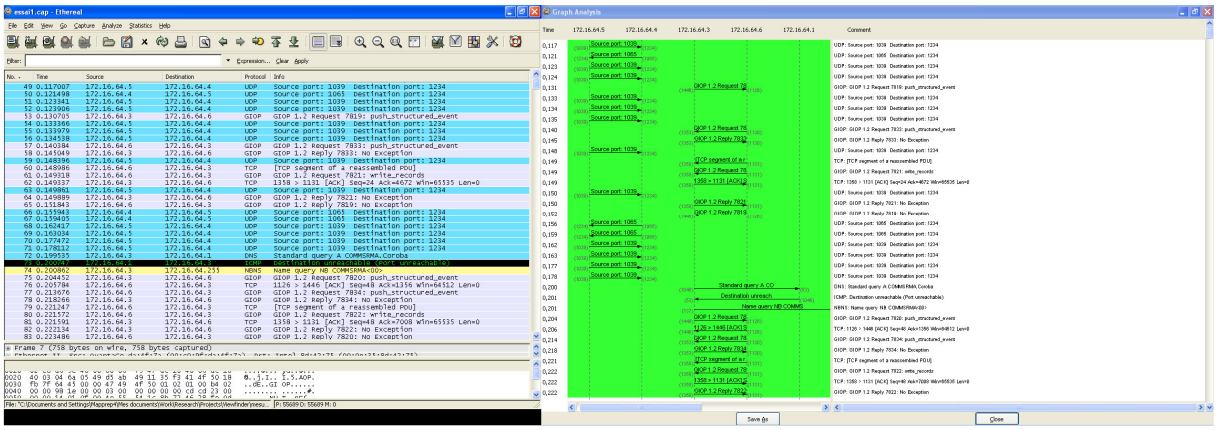

Figure 6. Delays in the control sequence

- As expected the standard Wi-Fi coms is limited to 30 m. At this distance the bandwidth is generally reduced to 1Mbs.
- The UDP communications are more robust than TCP communications to link variations.
- When the communication is at its optimal level, the control data flow (CoRoBA) requires a small bandwidth 0.5 Mbs for a 50 ms control period.
- The lack of priority between the different flows increases the latency in the control sequence.

Measuring WiMAX performance by IES

IES conducted a field test using WiMAX equipment to assess the performance of the technology with respect to latency, bandwidth and signal coverage.

The equipment used was the MacroMAX, provided by the Airspan company. This is a 802.16d-2004 compliant base station receiver, configured with multiple base station receivers to provide wireless broadband in urban areas to dozens of users simultaneously. The base station was connected to an omnidirectional high gain antenna. The subscriber station used in the test was composed of a WiMAX transceiver, an omnidirectional antenna and an ethernet port.

The testing site was located in a firefighting school, used for rescue mission training. Some particular characteristics of the location are the presence of a simulated chemical plant, a tunnel, an airplane, a ship and various buildings. Each of these sites is layed out as would be in a typical disaster scenario. The location was chosen because View-Finder is expected to be deployed under disaster circumstances. It is therefore important to assess the viability of wireless communication in this context.

A total of 7 sites were used for the tests. Site 1 was a Line of Sight (LOS) point near the base station, and meant to test maximum network capacity in ideal conditions. Sites 2 and 6 were inside a building and characterized indoor signaling capacity. Sites 3 and 5 tested Non LOS connectivity at greater distances. Site 4 tested signal strength inside a concrete tunnel, while site 7 was located behind a building and a thick concrete wall.

Maximum transmission throughput and end-to-end latency were measured for each site several times and then a weighted average was taken as the final result.

The software used for running the tests are Iperf [11], Netperf [12] and some minor custom programs.

Iperf measures maximum TCP/UDP/IP bandwidth, allowing the tuning of various parameters and UDP characteristics. It reports bandwidth, delay variation and datagram loss.

Netperf is a benchmark used to measure the performance of many different types of networking parameters. It provides tests for both unidirectional throughput and end-to-end delay. In our tests, we were concerned with the delay measurements.

Site	*Distance from base station (metres)*	*Description*	*Maximum bandwidth (kilo bytes / second)*	*End-to-end Latency (milliseconds)*	**Jitter (milliseconds)**
1	57	LOS[1]	370	23.62	9
2	63	NLOS[2], indoor, 1 concrete wall	257	23.37	9
3	168	NLOS, outdoor, behind enbankment, obstacles	212	26.21	11
4	240	NLOS, in a tunnel, behind 2 fire trucks	62	28.17	35
5	360	NLOS, outdoor	No signal	-	-
6	106	NLOS, indoor, several thin walls	180	26.04	15
7	130	NLOS, outdoor, behind building, additional thick wall	30	24	56

Table 1. Results of IES' WiMAX experiments.

End-to-end latency remains consistent at approximately 25ms even while other parameters vary greatly. This is several orders of magnitude higher than the latency incurred in fast ethernets, which is several tens of microseconds.

Bandwidth varies greatly even at short distances, depending on geographical factors and obstacles. Even though we did not include radio interference measurements in the experiment, it should be noted that it is also an important factor.

[1] Line-of-Sight.
[2] Non Line-of-Sight

Proposed solutions of integration

CORBA and Mailman are targeting different issues and their capabilities are complementary. However Mailman, as can be seen in the following table, is situated at a lower level than CORBA (Table 2) but also implements higher level services on top of the OSI model (Management of connection, discovery of servers, grouping, …) and consequently the combination is not straightforward.

	OSI Layer	**CORBA**	**Mailman**
7	Application (protocols to services)	CORBA IIOP	-
6	Presentation (network independent data representation)	Common Data Representation (CDR)	-
5	Session (reliability & & adaptation)	Implemented by the TCP stack when using TCP	Proprietary protocol
4	Transport	TCP, UDP	UDP
3	Network	IP	IP
2	Data link	Ethernet, Wireless	Wireless
1	Physical	Hardware	Hardware

Table 2. Situation of CORBA and Mailman in the OSI model

The nicest way of combining both systems should be to use only the part of Mailman managing the wireless communication but not the other services. This would not replace the default UDP stack implementation but supplement it with some info allowing managing message priorities. This could be done within the TAO CORBA implementation by using the pluggable protocols library [13].

A second possibility is to develop a CORBA component implementing the Mailman capabilities and being used as a single access point for the wireless communication.

Another solution should be to use Mailman as a gateway between the wireless access points. It means that CoRoBA components willing to send data over the wireless network would transmit data to the gateway. Of course this would imply that data be converted between some CoRoBA components and the Mailman gateway. This point would need to be further studied so as the use of the discovery service of Mailman with CoRoBA components. This solution would obviously add some extra loads for the on-board CPU and extra latency that would have to be evaluated.

Furthermore, the prioritization of the communication channels of CoRoBA by using the TAO Real Time CORBA implementation could offer a solution for managing message priorities.

Another approach is to reduce the amount of data being sent via the wireless communication by pre-processing the data on-board of the robot or by using multi-channels transmission hardware to separate the high priority signals from less urgent data.

Conclusions

A careful analysis is necessary when dealing with network communication, especially when data is sent through a wireless link. Having a common structure for all distributed components is mandatory in order to avoid incompatibilities between components developed by different partners.

As outlined by the results of the wireless experiments, bandwidth is quite limited compared to wired networks and can vary greatly depending on many factors. In a mobile wireless network such as that in View-Finder, this would constitute a bottleneck.
End-to-end latency in broadband wireless is much higher than in a wired network, so it is fundamental to add class based packet delivery to the View-Finder network.

Acknowledgements

The development described in this paper are supported by the **View-Finder FP6 IST 045541 Project.**

References

1. Eric Colon, Hichem Shali, Yvan Baudoin,CoRoBa, a multi mobile robot control and simulation framework, *Special Issue on* "Software Development and Integration in Robotics" of the International Journal on Advanced Robotics, pp 73-78,Volume 3, Number 1, March 2006.

2. Gamma E., Helm R., Johnson R and Vlissides J., Design Patterns Elements of Reusable Object-Oriented Software, Addison-Wesley, Professional Computing Series, 1995, ISBN 0-201-63361-2.

3. IEEE 802.11 Standards Group, IEEE 802.11e-2005—Medium Access Control (MAC) Quality of Service Enhancements, 2005

4. IEEE 802.16 Standards Group, IEEE 802.16-2004—IEEE Standard for Local and metropolitan area networks Part 16: Air Interface for Fixed Broadband Wireless Access Systems, 2004

5. G. Xylomenos and G. C. Polyzos—Link Layer Support for Quality of Service on Wireless Internet Links, Center for Wireless Communications, University of California, San Diego, 2000

6. D. W. Gage—Network Protocols for Mobile Robot Systems, Space and Naval Warfare Systems Center San Diego, 1997

7. L. Larzon, M. Degermark, S. Pink—UDP Lite for Real Time Multimedia Applications, Tech. Rep. HPL-IRI-1999-001, Extended Enterprise Laboratory, HP Laboratories Bristol, Bristol, UK, April 1999

8. J. Postel, RFC 768—User Datagram Protocol, ISI, 1980

9. Gill, C. & Smart, W. (2002). Middleware for Robots?, In Intelligent Distributed and Embedded Systems, Papers from the 2002 AAAI Spring Symposium, Gaurav S. Sukhatme and Tucker Balch (Ed.), pages 1-5, 2002.

10. Gowdy, J. (2000). A Qualitative Comparison of Interprocess Communications Toolkits for Robotics, Internal report CMU-RU-TR-00-16, the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA

11. M. Gates, A. Tirumala J. Dugan, K. Gibbs—Iperf, National Laboratory for Applied Network Research, University of Illinois

12. R. Jones—Netperf, The Hewlett-Packard Company

13. Carlos O'Ryan, Fred Kuhns, Douglas C. Schmidt, Ossama Othman, and Jeff Parsons, The Design and Performance of a Pluggable Protocols Framework for Real-time Distributed Object Computing

Middleware, (*updated October 14*) IFIP/ACM Middleware 2000 Conference, Pallisades, New York, April 3-7, 2000

14. Eric Colon CoRoBa, a multi mobile robot control and simulation framework, , Ph.D. Thesis, November 2006, Vrije Universiteit Brussel - Koninklijke Militaire School