Adaptive real-time video transmission over DDS

Marisol García-Valls, Pablo Basanta-Val, and Iria Estévez-Ayres
Distributed Real-Time Systems Lab.
Department of Telematics Engineering
Universidad Carlos III de Madrid
Avda. de la Universidad 30
28911 Leganés, Madrid (Spain)
{mvalls,pbasanta,ayres}@it.uc3m.es

Abstract- An increasing number of industrial applications include video processing capacities, which allow, among others, remote monitoring of industrial processes and control of private and public areas. Image processing has real-time requirements which result in resource demands at both node and network levels. Moreover, video is usually compressed and coded to be transmitted which generates variable bit-rate streams. This introduces variable processing requirements inside the node in terms of memory and processor cycles required for the processing of the sequence of different video frames. A direct impact on the network resource is also obvious since variable network bandwidth will be required to transmit the frames that may affect the bandwidth assigned to other streams. Efficient distributed video surveillance requires that real-time constraints be respected or, at least, quality of service guarantees (QoS) be provided. The traditional approach to video transmission has focused at the level of the network protocols. However, architectural solutions at the middleware level introduce higher flexibility and more efficiency in development time. This paper presents an architecture that precisely defines an integral set up of the different components that are relevant in achieving real-time and QoS-based video surveillance. The paper describes how the DDS standard for real-time distributed systems, can be used for this purpose; based on the decoupled interaction paradigm of DDS, higher complexity surveillance deployments are possible. A prototype surveillance system is presented which includes video transmission and adaptation to environmental sensing events.

I. INTRODUCTION

The benefits of image processing in industrial domains is evident. For example, it enables the control of the activity in private and public space. Moreover, it allows introducing a remote human eye in industrial processes which do not tolerate permanent human presence due to monetary cost or extreme environmental conditions.

In this context of remote process monitoring, image processing and transmission has real-time requirements in the sense that late image delivery is either annoying or of no use for the process to be controlled. However, in most industrial contexts, it is allowed that some images be dropped or delayed which introduces the concept of quality of service (QoS) management. In the field of computer networking, QoS refers to the resource reservation control mechanisms and statistical packet delivery, rather than to the resulting output quality [1,2]. However, in the real-time domain, where end nodes are mostly the main subject of study, QoS has been studied from the side of multimedia consumer electronics; in this field, QoS management is referred to as a trade-off between the assigned computational resources and the quality of the output [3,4,5].

The nature of video processing poses a number of significant challenges. One on hand, video processing relies on the usage of compression algorithms that generate variable bit-rate (VBR) streams. These algorithms produce different types of frames that may require significantly different amounts of computational resources. This introduces resource-demands variability in the nodes at both ends of the communication and also at network level. This is, for instance, the case of a sudden change in the type of incoming media that may require extra computational resources to absorb it. Such changes require an adaptation of the current structure of the system to a new configuration where the demands can be fulfilled neither threatening the system operation nor annoying the operator/user perception.

Moreover, industrial applications have become more sophisticated [6]. They have enriched the control of their traditional physical processes with human-assisted decisions based on video processing functionalities. Also, applications integrate heterogeneous hardware platforms and networks that offer a number of new facilities to users/operators. However, such new characteristics may introduce other sources of variability. This is, for instance, the case of applications that trigger execution by means of sensors that detect changes as, for example, human presence, position changes of some artifacts, luminosity changes, or temperature variations, among others.

All these sources of variations require that appropriate architectures be constructed that allow dealing with these heterogeneous triggers to offer the appropriate and expected response to the human operator in due time. Besides, efficient communication backbones will have to be combined inside these architectures to fulfill time requirements.

1 Version submitted to (and accepted for publication at) the IEEE 8th International Conference on Industrial Informatics. Osaka, Japan. 2010.
This paper presents an architecture for an intelligent distributed video surveillance system that is completely dynamic and reactive to environmental conditions. The architecture is based on real-time middleware solutions as [7,14] that allow the application to integrate decoupled components. Using a middleware solution brings a flexibility not known by traditional video and image processing applications which directly execute over network protocols and have very limited flexibility. Moreover, this specific architecture design which efficiently integrates a real-time middleware as the backbone component in the architecture allows offering true real-time support in the communication.

The paper is structured as follows. Section 2 describes the related work. Section 3 presents the interaction models of middleware with special attention to the publish/subscribe paradigm that naturally integrates decoupled interaction. Section 4 presents the architecture of the intelligent remote real-time distributed application and the interconnection among its different. Section 5 shows the adaptation of the architecture to integrate DDS standard as the core communication bus. Section 6 shows the implementation of the demonstrator to validate the architecture. Section 7 concludes the paper.

II. RELATED WORK

In the majority of industrial video surveillance systems, video processing is not integrated in a real-time framework where processing tasks are scheduled with some real-time protocol. Instead, the general approach is to use efficient operating systems (generally, Linux-based) that execute efficient image-processing and video-compression libraries; image transmission is also done over efficient transport protocols. Video transmission takes, traditionally, two main approaches:

- **network level**, that is efficient for transmission but lacks flexibility, or
- **middleware level**, that is flexible in terms of programming; however, without appropriate use, it is a source of inefficiency in resource usage.

At the network level, considerable research has been carried out in the past years [8,9]. Typical solutions are based on the stack type TCP/UDP over IP combined with RTP [10] (Real-time Transport Protocol) that allows transmission of real-time data over multicast or unicast network services. However, RTP does not address resource reservation and does not guarantee quality of service for real-time services. RTP is augmented with RTCP [10] (Real-Time Control Protocol) that allows monitoring of the data delivery in a manner that is scalable to large multicast networks to provide minimal control and identification functionality. RTSP [11] is an application level protocol for control over de delivery of data with real-time properties; it allows to control multiple delivery sessions, choosing delivery channels as UDP, multicast UDP, and TCP, and it provides means for choosing delivery mechanisms based on RTP.

Other approaches based on the usage of middleware techniques trade-off some efficiency for flexibility. This is, for instance, the case of [12] where a CORBA implementation based on TAO was used to offer real-time video transmission. Also, in [2], a CORBA based framework was presented to react to varying resource needs in video applications using the A/V Streaming service over CORBA.

Both approaches to real-time video transmission present drawbacks. Network level approaches are usually static solutions being basically a direct implementation over a transport protocol. In this case, a simple application update as adding/replacing a compression algorithm or a sensor data trigger may require re-architecting the whole application. On the side of the middleware approaches, CORBA-based solutions have been the ones proposed for real-time environments. CORBA is a very complete technology that presents a large number of interfaces for practically any type of required middleware functionality; however, it is a complex architecture that introduces execution overheads, specially if compared with other lighter weight technologies such as ICE (Internet Communications Engine) [13], DDS (Data Distribution Service for real-time systems) [14], or some specific Real-Time Java based solutions [15].

Therefore, existing approaches can be enhanced to offer appropriate support to the real-time nature of video surveillance which implies image processing and transmission with guarantees. Besides, new standards have appeared that enable the development of decoupled applications for video transmission which bring in flexibility at two levels. On one hand, the usage of a middleware is already flexible compared to direct implementation over the network level. On the other hand, using middleware solutions that offer QoS management allows to appropriately introduce real-time support to industrial surveillance.

III. BASIC INTERACTION MODELS IN MIDDLEWARE

A. Interaction Models

The selection of the main communication paradigm is of great importance in an architecture since the level of flexibility, decoupling, and efficiency depend on the underlying communication model.

Entities can communicate either synchronously or asynchronously. Synchronous communication may be done at clock-level (as in time-triggered communication or in schemes where, at least, logic clock synchronization is required) or at a higher-level based on RPC-like (Remote Procedure Call) communication. Asynchronous communication allows information exchange where there are no bounds on either process execution speed nor message transmission delays, and clocks can have arbitrary drift rates. Asynchronous communication can be implemented in two ways. On one hand, with events there is no need to have queue at the receiving end, so the communication fails if the
receiver is not available. On the other hand, by using messages, senders do not need to know the identity of the receivers but only of the message queue where the receivers are waiting. The middleware architecture related to this interaction model is MOM (Message Oriented Middleware).

How the different middleware technologies make use of the different interaction models and communication paradigms is important to efficiently implement the communication. Following, the basic middleware interaction models are presented: RPC, DOM, and MOM.

RPC provides a synchronous communication model based on invocation of code which resides in a remote machine in the form of a procedure (function or method, depending on the programming language used). This scheme is available on a huge number of operating systems. RPC-based middleware (such as Java RMI) provides three different services: definition of interfaces, marshalling and unmarshalling services, and network communication. Therefore, clients use the remote calls in the same way as a local call. However, a purely synchronized middleware as an RPC-based one offers low efficiency in large networks, and it only allows the development of tightly-coupled applications.

Distributed Object Middleware (DOM) was the next natural step of RPC. It brought in the facilities of object-oriented programming as encapsulation, support for inheritance, object references, and exceptions. However, it lacks scalability since object middleware may not always be applicable to non-object-oriented environments and programming languages. This is a synchronous scheme in its basic form; however, evolution of this model includes support for asynchronous communications.

Message Oriented Middleware (MOM) allows the communication of data stored in the form of messages. In the middleware messages are software structures that encapsulate data to be exchanged. MOM relies on the usage of message queues which provide asynchronous communication between sender and receiver. They have no natural support for synchronous communication directly between sender and receiver. MOM is usually used to provide event notification and requests for service execution.

B. The Publish/Subscribe interaction paradigm

Publish/Subscribe (P/S) interaction paradigm provides an asynchronous means for communication of specific data types. P/S architectures allow constructing a communication cloud among several nodes that are interested in specific data contents. Where data comes from and where it goes to is not the focal point. This is a powerful means for building decoupled systems. In P/S, there are publishers that are the entities that produce information and publish data in the system, and subscribers that are the consumers of such information generated in the system. Subscribers express their interest in a particular type of information by registering (subscribing) to it. Figure 1 shows a typical P/S system.

When a publisher sends information, an event is generated that contains the published information as a sort of payload. An event notification service is available in the middleware that implements this model to notify this event to the subscribers that, in this way, receive the information. As subscribers are usually interested in a subset of all events produced, they express their interests by issuing one or more subscriptions. Each subscription acts as a filter on the set of events produced in the system. The event notification service acts as the mediator between publishers and subscribers, completely decoupling the interactions among them. More specifically, P/S interaction paradigm is able to provide an interaction characterized by: space decoupling, time decoupling, and flow decoupling.

As a result, P/S paradigm offers a flexible interaction model in distributed applications, allowing the development of a complex system made of remote decoupled components that interact based on their interests for specific data contents. DDS standard implements this model of interaction, and it also offers specification of QoS parameters.

IV. ENHANCED MIDDLEWARE FRAMEWORK FOR REAL-TIME SUPPORT

Real-time networked surveillance systems require a middleware architecture which vertically offers support for timeliness in communicating images. Figure 2 shows the architecture of the middleware framework proposed. This architecture is general enough to allow for selection of different communication paradigms which are contained in the backbone layer in order to introduce different levels of support for timely video transmission and processing.

When a publisher sends information, an event is generated that contains the published information as a sort of payload. An event notification service is available in the middleware that implements this model to notify this event to the subscribers that, in this way, receive the information. As subscribers are usually interested in a subset of all events produced, they express their interests by issuing one or more subscriptions. Each subscription acts as a filter on the set of events produced in the system. The event notification service acts as the mediator between publishers and subscribers, completely decoupling the interactions among them. More specifically, P/S interaction paradigm is able to provide an interaction characterized by: space decoupling, time decoupling, and flow decoupling.

As a result, P/S paradigm offers a flexible interaction model in distributed applications, allowing the development of a complex system made of remote decoupled components that interact based on their interests for specific data contents. DDS standard implements this model of interaction, and it also offers specification of QoS parameters.

Fig. 1. P/S architecture

As a result, P/S paradigm offers a flexible interaction model in distributed applications, allowing the development of a complex system made of remote decoupled components that interact based on their interests for specific data contents. DDS standard implements this model of interaction, and it also offers specification of QoS parameters.

IV. ENHANCED MIDDLEWARE FRAMEWORK FOR REAL-TIME SUPPORT

Real-time networked surveillance systems require a middleware architecture which vertically offers support for timeliness in communicating images. Figure 2 shows the architecture of the middleware framework proposed. This architecture is general enough to allow for selection of different communication paradigms which are contained in the backbone layer in order to introduce different levels of support for timely video transmission and processing.

When a publisher sends information, an event is generated that contains the published information as a sort of payload. An event notification service is available in the middleware that implements this model to notify this event to the subscribers that, in this way, receive the information. As subscribers are usually interested in a subset of all events produced, they express their interests by issuing one or more subscriptions. Each subscription acts as a filter on the set of events produced in the system. The event notification service acts as the mediator between publishers and subscribers, completely decoupling the interactions among them. More specifically, P/S interaction paradigm is able to provide an interaction characterized by: space decoupling, time decoupling, and flow decoupling.

As a result, P/S paradigm offers a flexible interaction model in distributed applications, allowing the development of a complex system made of remote decoupled components that interact based on their interests for specific data contents. DDS standard implements this model of interaction, and it also offers specification of QoS parameters.

IV. ENHANCED MIDDLEWARE FRAMEWORK FOR REAL-TIME SUPPORT

Real-time networked surveillance systems require a middleware architecture which vertically offers support for timeliness in communicating images. Figure 2 shows the architecture of the middleware framework proposed. This architecture is general enough to allow for selection of different communication paradigms which are contained in the backbone layer in order to introduce different levels of support for timely video transmission and processing.

As a result, P/S paradigm offers a flexible interaction model in distributed applications, allowing the development of a complex system made of remote decoupled components that interact based on their interests for specific data contents. DDS standard implements this model of interaction, and it also offers specification of QoS parameters.

IV. ENHANCED MIDDLEWARE FRAMEWORK FOR REAL-TIME SUPPORT

Real-time networked surveillance systems require a middleware architecture which vertically offers support for timeliness in communicating images. Figure 2 shows the architecture of the middleware framework proposed. This architecture is general enough to allow for selection of different communication paradigms which are contained in the backbone layer in order to introduce different levels of support for timely video transmission and processing.
UDP/IP. Over this, the layer offers RTP enhanced with RTCP based communication at transport level that is suitable for real-time transmission. Also, RTSP can be offered for streaming, although usually UDP/IP is enough for the basic backbone core communication models.

The Backpack Communication and Resource Management Layer (BL) contains three main components:

- **Core communication models** which allow using any of the basic middleware interaction model that may exit: P/S, MOM, DOM, or RPC. This layer offers great flexibility inside the framework. If a synchronous communication middleware model is used, as RPC, the framework will export this paradigm to applications. If an asynchronous communication middleware is used, as one based on P/S, the framework will allow developing decoupled distributed systems.

- The **QoS Management** component aimed at providing real-time operation or, at least, QoS guarantees; the management of the communication and computation resources is carried out by the QoS management entity. This way, it is possible to guarantee the required resources and allocate the available resources to the application whenever there are changes in the requirements of streams. Avoiding over-reaction to transient changes is absorbed by the QoS management entity following a protocol based on [3].

- The **Custom Protocol Stack** provides real-time services (scheduling, real-time network protocols) for backward compatibility with legacy systems or for supporting applications that have hard real-time requirements. This part is mostly executed on custom dedicated hardware.

For the video surveillance system, the publish/subscribe (P/S) paradigm has been selected as core communications component. P/S allows for development of decoupled applications. The topic mechanism (present, for instance, in DDS standard) is flexible enough to allow transmission of video frames. Different streams may be supported by using different topics such as shown in figure 3.

To optimize the usage of the middleware framework when using a P/S backbone, integration of several streams into one topic can be done. We have used a tagging mechanism inside topics based on the usage of keys.

![Fig. 3. P/S communication structure through topics. Streams are mapped to topics](image)

V. INDUSTRIAL VIDEO SURVEILLANCE APPLICATION

The application is a prototype of an industrial real-time video surveillance system. It has been developed using DDS technology to provide operators with real-time video transmission from the remote camera sources. The demonstrator contains network cameras capable of capturing images, performing basic image processing and transmission with RTP. In our application, there are server nodes that receive and store the video captured by network cameras. Each camera sends video to the specified server node as established at configuration time. There are other nodes that also receive images but they do not apply any further processing nor store data. These are the operation nodes used typically by operators and/or security staff.

![Fig. 5. Structure of industrial surveillance application](image)

Figure 5 shows the video surveillance system deployment. In our particular case, there are two critical zones each of which contains two cameras pointing at different positions. In one zone, there are also sensors that detect unexpected motion and luminosity. Whenever an unexpected event is triggered by sensors, recording is triggered and cameras start transmitting video to server nodes and connected operation nodes. Video is transmitted in real time between cameras and nodes using RTP and between nodes using the middleware framework with a DDS communications backbone.

The middleware architecture framework has been easily extended, as seen in figure 6, to apply to the surveillance system.

![Fig. 4. Optimized P/S communication structure through tags where only one topic is necessary. Streams are mapped to topics](image)
The layers of the middleware are the two upper layers: Enhanced functionality contains sensing capacities, video control and configuration capacities, and resource claims (QoS) capacities. Backbone communication is based on P/S DDS technology.

To apply the real-time middleware framework presented in figure 2, small changes shown in figure 6 have been made. The middleware has been slightly modified in an easy way. First of all, P/S based communication core has been selected based on DDS technology. DDS standard is targeted at distributed real-time communication offering QoS parameters that allow controlling the communication. The application runs DDS over RTPS (Real-Time P/S Protocol) over UDP which is datagram-based and more efficient than TCP. Data coherency and reliability is controlled by DDS.

An Enhanced functionality layer (EL) has been added to offer support for the specifics of the video surveillance application. EL contains the following components:

- **Environmental sensing** that contains the sensing logic for data analysis and alarm detection. Interaction with sensors is also integrated with commands for setting/reading sensing parameters (mainly, sensed data type, sensing frequency, interaction paradigm as pull/push modes).

- **QoS specification** component allows specifying the requirements of the video transmission such as realiability factors, timeliness of the transmission, data liveliness, and historical information activation. Such parameters will, in turn, be fed to the QoS parameters of the DDS-based communication backbone.

- **Video configuration and video control** component contains the logic for configuration of the application setting such as the initial mapping of network cameras to server nodes, transmission modes, video formats, permission levels, and requesting video from specific cameras or zones.

VI. IMPLEMENTATION RESULTS

The deployment of the application employs real surveillance platforms and emulates the exact features of an industrial environment, which it is designed for. Near future plans include a real set up deployment in the context of the iLAND project.

Node heterogeneity of the application requires that different node types contain a special set of architecture modules depending on their resource capacity and requirements for external interaction. Sensors are programmed to detect movement and luminosity. Movement detection can trigger alarms of unexpected presence that activates recording of the corresponding cameras, deep zooming of some area, and compression adjustments to deliver high quality images. Monitoring of luminosity conditions triggers adjustments in video capturing conditions. Such behaviour is fully dynamic and adaptation is achieved by means of an active monitoring component in the sensor gateways. Therefore, sensors contain only the basic functionality to read, write, delete, and send parameters, for instance, setting the sampling rate.

Sensor gateways or sensor control nodes (for example, \text{n}_\text{ctrl} in figure 5) are the base station nodes that contain functionality to route towards the sensor network, and also they contain the logic for analysis of, and reaction to, sensor data readings. The component environment sensing contains such functionality. Figure 7 shows the easy of setting a threshold value to trigger temperature readings. Communication with the sensor could be done, for instance in Java-based sensors as the efficient Sentilla Perk, by means of sockets or even RMI. In current implementation, UDP has been used.

```java
/**
 * Parametric constructor
 * @param count Message count
 * @param moteId sensor id
 * @param temperature Temperature
 * @param isRequest is it a request or a response
 */
public SimpleMessage(long count, long moteId, double temperature,
Units code) {
    setCount(count);
    setMoteId(moteId);
    setTemperature(temperature);
    setUnits(code);
}

/**
 * Get the message count
 * @return the count
 */
public final long getCount() {
    return count;
}
```

The rest of nodes of the network implement practically the whole architecture. Server nodes (\text{n}_2, \text{n}_3, and \text{n}_4 in figure 5) and operation nodes (\text{n}_1 and \text{n}_5 in figure 5) contain all components except for the environment sensing, which is specific for sensor gateways, as explained above. Server nodes may be processing and recording a number of streams. The QoS Resource Management component performs run-time monitoring of resource usage, mainly CPU and memory consumption. The network link between the sending camera and the server is also monitored by this component; if required the compression factors will be altered in order to fit all required streams. Moreover, load balancing is also
performed at this component so as to change the server assigned to the storage of some video camera input in situations where overload conditions are detected.

In network and operation servers, the middleware architecture is implemented with a PC based infrastructure with dual core hardware, Linux OS, UDP communication over IP, and DDS communications backbone as core real-time infrastructure middleware. DDS core is used for transmission between the server node and the operation node; both nodes are shown in figure 8. Java-based display facilities are used for the user interface for video visualization and application configuration.

Figure 8 shows part of the real deployment with an operation node and a server node for video recording. From an optimal luminosity condition as shown in figure 9, a change on the luminosity conditions (caused by a human hand as shown in figure 10) is forced. Immediately, the image resolution goes down and the image size is reduced in the screen to offer appropriate visualization. This action is required (it is embedded in the video control component) since low luminosity generates unbearable image noise; therefore, the resolution and compression of the video must be altered.

Figure 11 shows a real execution to test the average delay times incurred by the current implementation of the architecture. It can be seen that the obtained times are on the order of hundred nanoseconds for transmission among a server node and an operation node for a packet size of 1000 bytes. This size is the typical one used by the video streaming surveillance system. The numbers obtained in the present configuration and with the specific DDS implementation used also shows that the core communications backbone is very efficient in terms of buffer management.

### VII. Conclusions

An increasing number of industrial applications include video processing capacities, which allow remote monitoring of industrial processes or remote control of private and public areas. However, image processing has real-time requirements which maps to being demanding in resource needs at node level and network level. Video is usually compressed and coded to be transmitted which generates variable bit-rate streams. This introduces variable processing requirements inside the node in terms of memory and processor cycles required for the processing of the sequence of different video frames. A direct impact on the network resource is also
obvious since variable network bandwidth will be required to transmit the frames that may affect the bandwidth assigned to other streams. Efficient distributed video surveillance requires that real-time constraints be respected or, at least, quality of service guarantees (QoS); therefore, the architectures of the solutions should define an integral set up of the different components that are present are integrated in a way which allows to provide QoS-based video transmission. Traditionally, specific transmission protocols are used for video transmission which very low-level programming and little flexible. In this paper, we present how real-time middleware can be used for this purpose in a way that allows to have more flexible and efficient deployments. We have showed the easy of adaptation of the architecture to different node types. A P/S middleware based on DDS standard has been used as the core communications backbone which introduced QoS guarantees at middleware level. Also, average delay times in transmission have been presented to show the efficiency of the approach.

REFERENCES


ACKNOWLEDGEMENTS

This work has been partially funded by the ARTEMIS Call1 project iLAND (middLewAre for Deterministic dynamically reconfigurable networked embedded systems, ARTEMIS-JU 100026) and ARTISTDesign NoE (IST-2007-214373) of the EU 7th Framework Programme. Also, authors would like to thank most Iago Rodríguez and Laura Fernández for their implementation help.