RTPS middleware for Real-Time Distributed Industrial Vision Systems

Basem Almadani
Institute for Automation in Montan
University Leoben, Austria
Al.madani@unileoben.ac.at

Abstract

Designing and constructing Real-Time Distributed Industrial Vision Systems (RT-DIVS) from scratch is very complicated task. RT-DIVS has Conflicting requirements such as reasonable development cost, ease of use, reusable code and high performance. The success key in building such systems is to recognize the need for middleware software. Middleware plays a major role in developing distributed systems efficiently. Real-Time Publish-Subscribe (RTPS) model is one of the latest developments in Real-Time middleware technologies. Network Data Distribution Service (NDDS) is RTPS middleware developed by Real-Time Innovation (RTI). NDDS is widely used in Real-Time distributed and embedded systems for mission critical applications. The research work presented in this paper discusses the employment of NDDS for RT-DIVS and the advantages of NDDS's Quality of Service (QoS) policies in covering the requirements of RT-DIVS. An experimental test set-up is used to verify the NDDS's performance for RT-DIVS. Tests results show that RTPS middleware (and NDDS specifically) is suitable for soft and firm timelines requirements for distributed industrial vision systems.

1. Introduction

Industrial vision systems are employed to perform different functions such as quality assurance and logistic management. Those systems have usually multiple cameras connected with sensors, actuator and PLCs via a network. Image Processing (IP) task in RT-DIVS must maintain a set of time constrains. There are three types of time constrains in RT-DIVS: image acquisition, image transportation and image processing time. Time constrains in Real-Time systems are specified by a limit as in Hard Real-Time systems or as a range of time. The time constrain range can be long as in Soft Real-Time systems or short as in Firm Real-Time systems [1]. This paper discusses firm timelines requirements for image transportation in RT-DIVS.

In the last two decays, different middleware technologies are used to build distributed Real-Time systems for industrial applications. Middleware types differ mainly in the communication model and in the infrastructure hardware and software technologies used. The most famous middleware types are:

- Transaction processing middleware (TPM): depends mainly on transactional database.
- Object Request Brokers (ORB): such as CORBA and DCOM.
- Remote Procedure Calls (RPC)
- Message Oriented Middleware (MOM)

Each of these types has specific features that suit certain applications. MOM is the most suitable middleware for RT-DIVS because it support events management and match the needs for distributed systems as will be shown in the following section. MOM has two categories, message queuing and message passing middleware. Publish-Subscribe (PS) is a message passing middleware.

Middleware exchange information between components in different communication models such as:

- Synchronous client/server model: reliable request-reply oriented with flow control to avoid network congestions.
- Asynchronous model: data sender pushes data over the net and assumes reliable receivers on the other end.
- Fan-Out model: central process or server sends data to multiple clients.
- Fan-In model: Multiple processes send data to central server.
- Point-to-point (PTP) model: connection oriented single sender and single receiver model.
- Many-to-many model: can be seen as a combination of different models mentioned above where several processes or systems are exchanging data between each other [2].

The implementation of communication model in certain middleware depends on the network protocol used and the application design. In Ethernet networks, the transport layer has two main protocols, namely TCP/IP and UDP/IP. Ethernet networks are considered to be statistically Real-time networks. TCP/IP protocol is asynchronous connection oriented and reliable protocol. TCP/IP is not designed for Real-Time applications because it is not deterministic. UDP/IP is asynchronous user Datagram protocol, which is faster than TCP/IP but reliability is not assured while no handshaking mechanism is implemented. Studies showed that UDP/IP could perform well for a wide range of Real-Time applications in industrial environment. Additional services in the UDP/IP based

middleware can assure reliability for mission critical systems [3].

In this paper, the employment of Real-time Publish-Subscribe (RTPS) middleware for RT-DIVS is discussed. Network Data Delivery Service (NDDS) is RTPS middleware developed by Real-Time Innovation (RTI) and widely used for mission critical applications such as defense projects. This research is aimed to find how suitable RTPS for RT-DIVS for quality assurance and logistics applications in steel industries. The long-term objective of this research is to build an infrastructure platform for RT-DIVS developers to ease design and construction process for those systems [4].

2. RT-DIVS Requirements

There are common requirements for RT-DIVS. This section highlights some of those requirements with special focus for quality and logistics vision systems in steel industries.

- Soft and Firm Real-Time requirements for image transportation (excluding image acquisition time) with maximum latency of 35ms for 64KB image size and 20-50 images per second rate on dedicated or fire-wall protected network.
- Reliable image transportation and detection of any dropped frame.
- Detecting new cameras and network components that joined the network and add them dynamically to live components list. Also detecting disconnected or unreachable components and remove them from live components list along with generating proper log and alarm signals.
- Controls over resources such as maximum memory buffers for senders and receivers with low and high watermarks indications.
- Online configuration for data senders and receivers to inform network components about new data sources.
- Configurable data communication model. RT-DIVS needs to deal with some components in client/server manner with certain pooling mechanism. The developer should be able to choose between request-reply model and Publish-Subscribe model when dealing with network components.
- Distributing data according to certain criteria. Vision systems used to have smart cameras where the image is partially processed (e.g. extracting the region of interest) before sent over the net. It is important to send images over the net to receivers those are able to use the images. Receivers should be able to specify certain criteria for the required images. Some quality systems need to report only defects in the watched material by vision system.
- The developer should be able to separate the applications logically over the net to simplify debugging and troubleshooting processes. The receivers of certain function should not receive all the images generated over the net but only those images related to its function.
- RT-DIVS should be cost efficient and easy to modify. Using Real-Time operating systems such as QNX and VxWorks raises the development cost dramatically. Ada95 programmers cost is much higher than C/C++ programmers. Common of the shelf (COTS) technologies should be used where suitable to reduce the need for specialized consultants. It is recommended to use popular OS, such Windows, Linux and Unix, and programming languages in order to minimize the development and maintenance costs.
- RT-DIVS should be adaptable to suit a range of application domains. Developing application specific solution is a high cost practice. The solution should be configurable to suite changing requirements in industrial environment. Only image-processing algorithm should be system specific for certain application, other features should be configurable as far as possible.
- The cost of adding new component, in terms of the code change and configuration time, should be kept the lowest possible limit.
- RT-DIVS should be able to exchange information with systems based on different platforms, OS and programming languages [5].

3. Employing NDDS for RT-DIVS

This section describes basic NDDS features[1] and those QoS related to RT-DIVS and how they are employed to support in design and construction of RT-DIVS.

3.1. NDDS features and QoS

NDDS is a Real-Time Publish-Subscribe message passing middleware uses UDP/IP protocol. Each data source over the net is defined as a Publisher. Each publisher can publish one or more topic. New issue is a combination of topic, type, issue-identification and user data. Network applications can subscribe for one or more topic. Subscribers will receive copies of new issues generated by publishers. Publishers and subscribers don't need to know each other by name or IP address. Publishers don't need to know any thing about subscribers; they only care about generating correct data. Subscribers don't care who generated the new issues over the net. Each publisher has certain strength for each topic. If there is more than one

[1] Full NDDS description and features can be found in www.rti.com

publisher of the same topic over the net, subscribers accept issues from the publisher who has the highest strength value. If that publisher did not generate new issues for certain time window, subscribers accept issue from other publishers [6]. The following list includes the most important QoS in NDDS those are related to the scope of our work.

- Deadline: Subscribers can define a deadline period, which present the maximum waiting time the application can wait for new issues. The application will be notified after deadline limit and an exception will be raised.
- Durability: which specify storing mechanism for new issues. The first possibility is to deliver new issue to those subscribers on the net at the time the issue is published and no issues are stored. The second possibility is to store new issues to serve late subscribers join the network after the time of publication. The amount of issues (in terms of memory size) and the memory type (temporary or permanent) are two configuration parameters.
- Latency budget: this is a hint about the maximum acceptable delay from the time the issue is generated to the time it is delivered to subscriber. This information is important for the middleware to optimize its internal operations. Publisher and subscribers are not notified about this information.
- User data: includes some information that can be useful for the applications such as security credentials or any data decoded for the application.
- Reliability: this policy indicates the level of reliability the application wants or the publisher assures. Reliability can be Bets-Effort or Reliable. This policy depends on History and Resource-Limits policies.
- History: specifies the amount of issues to be stored. History can be KEEP_ALL or KEEP_LAST with certain depth to specify the number of latest issues to keep.
- Resource Limits: to control the amount of resources the middleware can use to satisfy application or QoS needs.
- Presentation: describe the coherence and the order of the issues when presented to subscribers [4].

3.2. NDDS objects and functions for RT-DIVS

To explain how NDDS helps in building RT-DIVS, NDDS objects and functions for each system component are described.

Measurement Node (MN), which is a network camera, should publish a *registration* topic to identify itself to the Measurement Server (MS) who is subscribed for registration topic. Registration topic's strength should be the same for all MN. Registration topic includes information about the camera type, frames rate, frame size, and its function. MN publishes Measurement Image (MI) topic, Calibration Image (CI) topic, Status topic and watch topic. Watch topic is used by MS to assure that MN is connected to the network without polling for MN. Topics names (exclude registration topic) include a postfix of the MN ID (e.g. Watch_MN201, Status_MN201, MI_MN201 and CI_MN201), where MN201 is the MN identification number. MS should publish Registration Confirmation (RC) topic to MN only once in each registration process. MN should subscribe for RC_MNId topic, with specific deadline, immediately after publishing the registration topic. If the deadline for RC is expired, MN will publish a new registration issue until the MS is up, or back to the network, and a registration is confirmed.

On the other hand, MS keeps a list of registered MN, and any other active network components if any, and updates the status for any MN that didn't send an issue of its watch topic (Watch_MNId) within deadline period. MS publishes *Calibration* topic for each MN (e.g. CALIB_MNId) includes measurement and device calibration instructions. MN should subscribe for calibration topic after receiving RC_MNId issue. MS publishes a Command topic (e.g. COM_MNId), which includes certain instructions such as restart MN or flush history to remove old images from MN temporary memory if there is one. Issue ID should be checked in MS to assure that images are in sequence if high reliability is required. Reliability setting depends on the parameters mentioned in the previous section, which differ upon the application requirements.

Two image acquisition modes are required in RT-DIVS, streaming and request. In streaming mode, MN should publish new MI issue at least once in each deadline period specified in the publication properties for that MN and NDDS will raise an exception to MS otherwise. In request mode, MI issues are generated when MN receives a signal from position sensor for example or measurement command by MS.

4. Experimental work

The aim of this part in the research is to test NDDS capabilities in transporting images with different sizes and frequencies under several configurations. The objective is to measure image transmission time between two computers without clocks synchronization as recommended by several middleware vendors [7]. In order to do that, the image is transmitted from one computer to another and bounced back to its source. A timestamp is attached to the image and the round trip time is calculated when the image is back. The test software was programmed in Java and we believe C or C++ programmes can achieve a better performance.

An initial test was performed based on:

- 10KB image size with 35 image/second (35Hz)

- Number of images transmitted = 4414
- Java processes were given Real-Time priority on WS2000.
- Forced garbage collection on both computers.

The following results was obtained:

- Test duration=126081ms
- Max. Round Trip Time (RTT) = 150ms
- Min. RTT= 10ms
- Mean RTT= 25.6590ms
- Median RTT= 20ms
- STD RTT= 17.2987ms

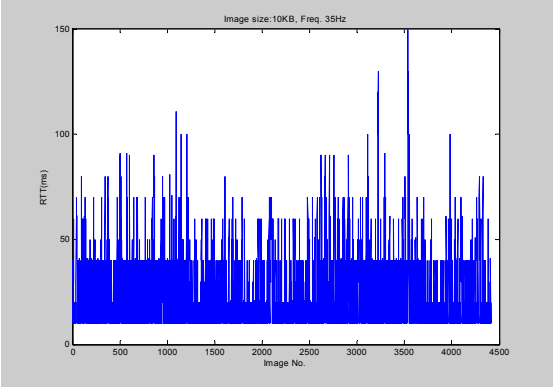

Figure (1): MATLAB Plot for RTT

RTT (ms)	Images	%
10	1635	37
20	918	20,80
30	440	9,97
40	819	18,55
50	168	3,80
60	136	3,08
70	82	1,86
80	26	0,59
90	10	0,23
100	7	0,16
>100	5	0,11

Table 1: RTT analysis

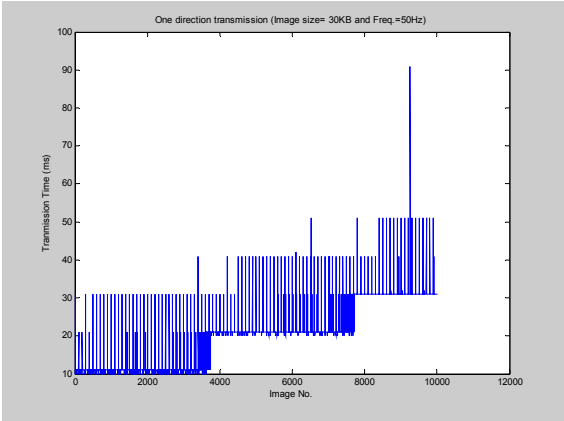

Figure (2): One direction transmission.

The second test was for:

- 30KB image size with 50Hz image rate.
- Image transmission in one direction.

The obtained behavior is shown in figure (2). With buffer size tuning, a more deterministic behavior was obtained as in Figure (3).

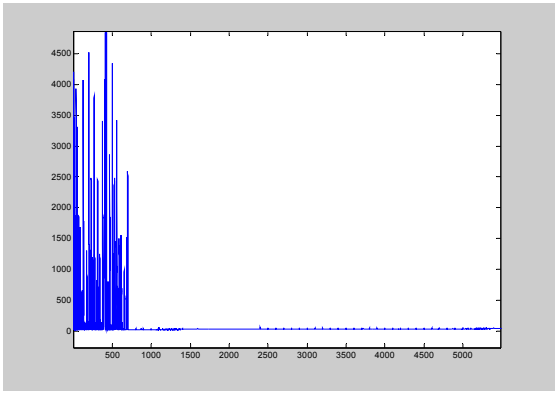

Figure (3): Tuned one direction transmission.

The time difference between the two clocks should be around to represent stable behavior. The behavior in the firs 800 images is not stable and the reason is not obvious for us yet but tuning and configuration work continue.

5. Conclusion

The implementation of RTPS in NDDS middleware is suitable for soft and firm timelines requirements in RT-DIVS and specifically for quality assurance and logistics management applications. NDDS simplifies the design and construction phases for distributed vision systems and the overall system's development and maintenance cost is reasonable. NDDS helps to build a reusable system that is configurable for wide range of industrial applications with low modification effort. Additional test are planned to achieve better and more deterministic performance.

References

[1] Bruce Powel Douglass, "Real-Time UML", ADDISON-WESELY 1998

[2] Talarian Corporation, "Everything you Need To Know About Middleware", 2000

[3] Real-Time Innovation (RTI) Inc., "Build-Your-Own Middleware Analysis Guide", 2001

[4] Object Management Group (OMG), "Data Distribution Service for Real-Time Systems Specification", 2003

[5] D. AI. Wu, L. Guan, G. Lau and D. Rahija, "DESIGN AND IMPLEMENTATION OF A DISTRIBUTED REAL-TIME IMAGE PROCESSING SYSTEM", Proceedings of the 1st International Conference on Engineering of Complex Computer Systems (ICECCS'95)

[6] Real-Time Innovation (RTI) Inc., "NDDS Getting Started Guide", 2002

[7] Talarian Corporation, "Guidelines for Evaluating Middleware Products", 2000