Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 25 (2014) 385 – 392

www.elsevier.com/locate/procedia

8th International Conference on Digital Enterprise Technology - DET 2014 – "Disruptive Innovation in Manufacturing Engineering towards the 4th Industrial Revolution

Evaluating a prototype approach to validating a DDS-based system architecture for automated manufacturing environments

M.S. Essers*[a], T.H.J. Vaneker[a]

[a]*University of Twente, Enschede, Laboratory for Design, Production and Management, Department of Engineering Technology, Drienerlolaan 5 7522 NB Enschede, The Netherlands, Tel. +31-53-489 25 20, E-mail: m.s.essers@utwente.nl*

* Corresponding author. Tel.: +31-53-489-3192. *E-mail address:* m.s.essers@utwente.nl

Abstract

Data Distribution Services (DDS) are emerging as communication systems in manufacturing environments. One of the key features of a DDS based system is the ability to regain performance levels after the introduction or removal of a DDS participant. In implementing a DDS participant to an existing system, message transport speed and message latency is often sacrificed due to protection problems in OEM software. Validity and suitability for integration of OpenDDS specifically, a manufacturing system is evaluated by defining two implementation scenarios; a flexible approach with a dedicated DDS participant application, and a high speed approach integrating the OpenDDS API directly in the target application. The system is validated by monitoring performance, efficiency and robustness in use and implementation. This result is part of a system architecture, developed for project Smart Industrial Robotics (SInBot), that focuses on maximizing the efficient use of mobile industrial robots during medium sized production runs.. This modular system architecture is based on distributed intelligence and decentralized control to enable online reconfiguration of industrial robots in manufacturing facilities.

Peer-review under responsibility of The International Scientific Committee of the 8th International Conference on Digital Enterprise Technology - DET 2014 – "Disruptive Innovation in Manufacturing Engineering towards the 4th Industrial Revolution"

Keywords: Data Distribution Service (DDS); Industrial Robots; Manufacturing

1. Introduction

Current industrial robots play an important role in performing repetitive production tasks. They perform their tasks cost effectively and accurate over longer periods of time. The European production industry is moving toward higher added value production that must be lean and flexible in order to survive in a competitive market [1]. Industrial robots need to get an even higher intelligence, collaborative and multipurpose deployment and effortless transfer of processes by which one-off or limited series can be produced.

SInBot (or Smart Industrial RoBotics) [1] is a project that specifically targets composite machining tasks in the manufacturing environment. Where these tasks are now performed by million euro lathe and milling machines, SInBot sees an important role for industrial robots. The focus of project SInBot is on the development of methods and tools for simplifying definition, selection and use of intelligent, decentralized cooperation of industrial robots for manufacturing purposes. Work preparation for industrial robots is still a long and tedious task, containing a multitude of optimization iterations. When the words 'work preparation' are used in this paper, it is used to describe the process to generate manufacturing data for some hardware to create a product from its technical product data (e.g. CAD file and a product data file) as efficient as possible.

The time consuming work preparation, complexity and diversity of programming languages particularly present a problem for industrial robots and their implementation in small and medium sized manufacturing enterprises (SM/MEs). A common need for European SM/MEs is found in flexible (or agile) manufacturing, mass customization and a decrease in setup and programming times. In project SInBot, the current work preparation process is perceived as being too complex and containing too much iterations, where possible solutions lay in task-level programming and communication.

Peer-review under responsibility of The International Scientific Committee of the 8th International Conference on Digital Enterprise Technology - DET 2014 – "Disruptive Innovation in Manufacturing Engineering towards the 4th Industrial Revolution"

doi:10.1016/j.procir.2014.10.053

Many SM/MEs already use automated manufacturing cells with CNC Mills or Lathes. In these automated milling centres, the automation in work preparation is far more advanced than in current automated industrial robot cells. Milling stations are expensive machines with relatively small work envelopes, but ensure high accuracy and the ability to be programmed offline for near 100%. Since programming files, product data, and process parameters can be stored for later use, and the machine adheres to the offline programming (OLP) almost entirely, subsequent reorders are available for relatively low additional (reprogramming and reconfiguration) costs.

For project SInBot to achieve the foreseen steps in automation of work preparation, a flexible robot-based machining cell has to be able to translate CAD files fluently into manufacturing tasks. Until now, robots are controlled on a low level. Although several higher level programming languages and allow humans to give commands on a relatively high level, are starting to emerge. SInBot will use these high level languages as task descriptions in a distributed and decentralized system. As task descriptions become more abstract, specific and individual unit control is lost. Algorithms that deal with high level tasks, should provide ample input for the robot to perform its task efficiently, while being robust to ensure continuity.

The lack of accuracy of industrial robots is a problem in automating work preparation. A common method of improving the accuracy (i.e. moving accuracy closer to the repeatability), is to introduce external end-effector tracking sensors. Project SInBot has also identified positioning and trajectory accuracy as the main problem with industrial robots in machining composites. For any piece of machinery to be programmed offline, it must perform the planned manufacturing program predictably. Currently, industrial robots lack the predictability (accuracy) for efficient OLP. As soon as the industrial robots adhere to OLP according to industry standards (i.e. accuracy approximately as good as their repeatability), the work preparation paradigm can be truly addressed. Project partners are working on a short loop for tool trajectory and position correction. The remainder of this research is performed assuming industry standard OLP adherence of industrial robots, and will discuss the envisioned SInBot manufacturing system.

2. Problem Scenarios

A distributed approach to controlling manufacturing environments with industrial robots can be beneficial when roughly two criteria are met: 1) the manufacturing tasks extend over the work envelope of a single industrial robot, and 2) the manufacturing tasks encompass small series of identical products. These environments are the most interesting target for the SInBot decentralized, distributed system. Small and medium sized enterprises are also situated in this scenario, since these normally produce in small to medium production series; often specialized parts (e.g. composite car roofs for high end sports cars). Additional factors that will impact the suitability of the SInBot system are 1) the need for different types of robots, featuring different performance specifications, 2) the need for communicating entities (other than industrial robots), and 3) the need for extendibility. Also note that manufacturing environments are evolving from mass-production to mass-customization, and even to personalization [2]. This evolution strengthens the position of the SInBot system. Manufacturing environments that are considered suitable for the SInBot system to excel, can be examined for distributed approach compatibility by looking at these production planning paradigms:

Paradigm 1: Two separate robots are working on different products, at different locations (Figure 1a). There is no interaction, nor interference between the two robots.

Paradigm 2: Two robots are working simultaneously on the same product (Figure 1b). There is interference, but no interaction. The current work preparation approach will take disproportionally longer due to the interference.

Paradigm 3: Two robots are working cooperatively on the same product, while the robots are online reconfigurable (Figure 1c). There is both interaction and interference. In the current work preparation approach, the interference and interaction must be implemented manually for all configurations of robots, for each product. The dynamic nature of this paradigm would allow too much configuration options for the current work preparation approach, and would render it unfeasible.

Paradigm 4: Two mobile robots are working cooperatively on the same product with a human worker (Figure 1d). There is both interaction and interference, from both robots and human workers. The inherent dynamic nature of paradigm 3 makes the current work preparation approach unfeasible, but introduction of human workers and the ability of the robots to co-operate, makes the current work preparation approach simply impossible.

While the first two paradigms are relatively common in the manufacturing industry, the latter two are not. However, both paradigm 3 and 4 are the focus of many studies to improve the efficiency of deploying industrial robots. For each of this paradigms, the SInBot system compatibility is examined.

3. Solution Context

SInBot's perspective on solving the SME problem is to introduce a flexible, mobile, plug-and-produce system in the manufacturing environment. Current manufacturing environments are static, rigid and established systems, consisting of (sub-)systems and software that is interfaced at the moment of purchase through (e.g.) socket or bus connections, and require extensive redesigns for each future upgrade [3].

Important advantages in the development phase obtained through this decentralized and distributed approach, is the possibility to develop module-based. The major challenge is to design, develop, and assemble a system that facilitates the modules, performs as demanded in speed and reliability, and supports plug-and-produce connection of both software and hardware modules. The SInBot manufacturing system proposal can be found in [4], and roughly consists of a communication layer capable of plug-and-produce connections from a variety of modules (Figure 2). The different modules can be separated by their goals; to facilitate

and to produce. Both categories can contain hardware (interfaces) and software. For example, *facilitating hardware* can be an AGV capable of reconfiguring production lines or cells, or a sensor network that publishes locations of known objects in 3D space. *Manufacturing software* may be a CAD-CAM software system, either automated or through manual information insertion into the system. The top layer of Figure 2 contains processing modules that are required for the system to function (production enablers), such as the CAD-CAM software, a capability mapping module, and an auctioneer. The bottom layer of Figure 2 contains hardware and software that enable the system to perform (performance enablers). Each module can be seen as enabling production or enabling performance. A production enabler can for example be swapped, as long as the replacement module produces the same output based on similar input. The production enabler is necessary for the system to function. A performance enabler can for example be removed without serious consequences to the functioning of the system, yet may influence production speed or quality. The system is far less dependent on performance enablers. Do note that a similar system can be built with large function overlaps between modules, essentially combining enablers of production and performance, generating a more flexible yet comprehensive and complex system.

In the envisioned system, the data-driven communication layer is a vital piece. This layer needs an accessible and well-performing data transport method for the flexibility and plug-and-produce facets to emerge. Within the subgroups that were discussed earlier, many new divisions can be perceived. There are entities that only share their information, and entities that only retrieve information. The more intelligent entities always both share and retrieve data. An example of such an intelligent entity would be an AGV from the facilitating hardware category; it requires positional data and tasks, while sharing status, progress and auction bids. An example of an entity that only shares information, but not retrieves any, is a sensor network. These subsystems share perceived data regardless of feedback. These examples are pictured in Figure 2 as devices that interface to the communication layer, in which new devices should be easily added later in the system life-cycle. The software in the communication layer should therefore be flexible, fast, reliable, and fault tolerant. The most important aspect, is that the system supports plug-and-produce interfaces for (sub-) system introductions. These are the key performance indicators on which to base the selection of a middleware communication system for the SInBot Manufacturing System.

Amongst the rising stars in the middleware communication software segment are Data Distribution Service (DDS) software providers. [5] DDS provides communication services through the publish-subscribe protocol in real-time and embedded systems. The general idea is that DDS participants can publish data regardless of who is listening, while other DDS participants can read data regardless of who is publishing it. Specific properties of the DDS participants can be set by changing the Quality of Service settings (QoS). QoS settings for example, can determine whether the DDS system will ensure that the data is either the last known published data set, the full published set (history), or something in between. By doing so, DDS introduces a virtual Global Data Space where applications can share information by simply reading and writing data-objects addressed by a topic and a key. The primary discriminator between DDS and other data-centric approaches is the Quality of Service (QoS) policy set, generating a dynamic, tuneable and scalable system [6-8]. Summarized, DDS enables fault-tolerant, dynamic, high speed complex transport of data. [3] These systems have proven to contribute to efficiency, flexibility and robustness in military, infrastructure, and other data-centric systems. Even though the promises are diverse, they have been scarcely implemented in manufacturing environments or projects that do so. [9]

In the manufacturing sector, there is a strict division between message-oriented and data-centric (publish subscriber) middleware (DDS) [10], in that the loosely coupled data-centric approach avoids complicated linking of applications and bottleneck databases. In both message oriented middleware [11, 12] and data-centric centric approaches, research explains the advantages these systems will bring to manufacturing systems, or the requirements for these systems to benefit from such approaches. For example, data-centric systems are valid when a) participants are distributed, b) interactions are data-centric instead of object-centric, c) dynamic nature of entities require predictable delivery, d) processes may be dependent upon the predictability of data delivery, or e) storage of data is local [13]. If, in addition to more than one of the aforementioned aspects, the system is in need of real-time availability of data, DDS offers the ideal solution. [14] Since project SInBot adheres to these aspects, DDS is chosen as the communication layer (see Figure 3). The validity of such a system will be proven with demonstrator systems, of which each subsequent system increments closer to a real-world implementation of the SInBot manufacturing system.

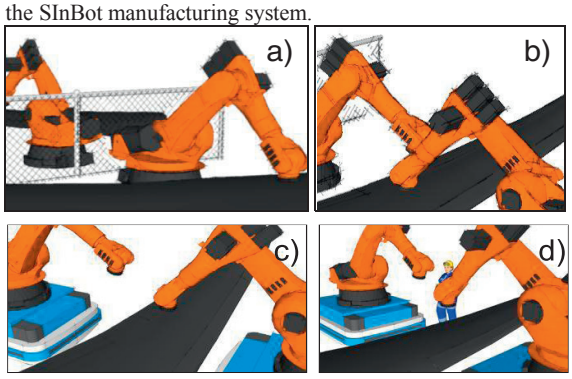

Figure 1: Industrial robot production planning paradigms.

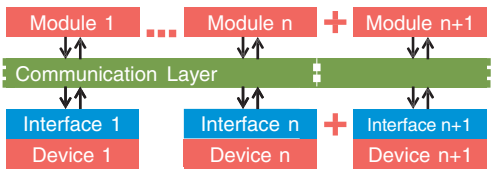

Figure 2: SInBot system abstract.

The hardware of these prototype systems are scaled up to the final demonstrator. The last demonstrator will show the validity of the system on an industrial robot and provide the last parameters required to prove the validity of a real-world implementation. The rule of thumb for each demonstrator, system, subsystem, or component, is that when Open Source (free) software and hardware can be proven to perform as is stated in the project SInBot goals, it is agreeable to conclude that the system would also function and perform similar or better, with professional, industrial hardware and software.

4. DDS Evaluation

4.1. Selection of a DDS system

The SInBot Manufacturing system is proposed as containing a data-driven DDS-based communication layer. However, the prototype is a proof of principle; a DDS system in the prototype would be beneficial, but not a hard requirement. (Proprietary) DDS systems were examined and compared to evaluate their compatibility in the framework, but emulating the layer or using a similar (yet simpler) system that adheres to the proof of principle is also viable. Unmistakably, a DDS system implementation in the early stages of the SInBot system development, specifically in the prototype, would reduce further development costs and time. Consequently, four major DDS players were examined on three categories: Performance, Application, and Liveliness (see Figure 4). At the time of writing, multiple DDS implementations are viable contenders. OpenDDS, CoreDX DDS, RTI Connext, and OpenSplice were selected to evaluate, knowingly ignoring MilSoft DDS, InterCOM DDS, and ETRI DDS due to a smaller supporting community. The selection criteria reflect the requirement of the middleware communication layer of the SInBot Manufacturing System, but also the consequences of building a prototype with these middleware solutions.

From building the SInBot Manufacturing System and the prototype, emerged the need for either a small or nonexistent license fee for Educational and R&D use of proprietary software. All the categories are evaluated with the most complete software version that have acceptable development fees per license. In most cases the commercial fee is unacceptable for the SInBot development phase; for these systems, the Open Source Community or Education version were used. The ‘Application’ category focusses on the implementation of the software system, evaluating the level of complexity measured in the required prior knowledge of the implementer and the indirect support the providers offer. The ‘liveliness’ category evaluates the update policies of the software provider (i.e. the update polices on their Open Source or Community editions) and the size of their support community. The speed doesn’t differ much between DDS software providers, the OMG compliance only slightly.

For emulation of a flexible communication layer, an easy implementation option would be to define a central (e.g. MySQL) database containing all information in an orderly fashion. Interfaces to such a database are easily written in a variety of languages from a variety of platforms. This example of a web-based emulation of a flexible communication layer is prototype-only, since a real-world implementation would be slow, unreliable, and create problems of centralized control systems in a distributed architecture. If failure occurs in any entity that shares information, or in the database itself, it would translate to productions faults, safety issues, and machine damage. Alternatively, some specific control systems provide a communication layer also; e.g. the Robotic Operating System (ROS). Next to multiple libraries and tools to aid development and prototyping of robots, ROS contains a publish/subscribe communication layer. This particular system works with a main node, containing information about the DDS participants, not unlike some other data-centric systems [15]. There are extensions to auto-discover the main node (master), maneuvering ROS’ communication method close to DDS systems using automatic topic discovery. For its all-round performance, OpenDDS was chosen for the SInBot manufacturing system. *DDS Interfacing*

DDS software providers promise fast communication between publishers and subscribers. However, for these latencies to be available to subsystems in a DDS-based system, the DDS API must be implemented directly in the software. A DDS communication entity is set up by a DataReader / DataWriter and a subscriber / publisher. A publisher or subscriber can contain more DataWriters or DataReaders. In this principle lay opportunities for fast communication and flexibility. However, this also means that each communication application would become unique, and rigid in deployment. Consequently, there is a potential problem if the publisher / subscriber application is incompatible with existing software.

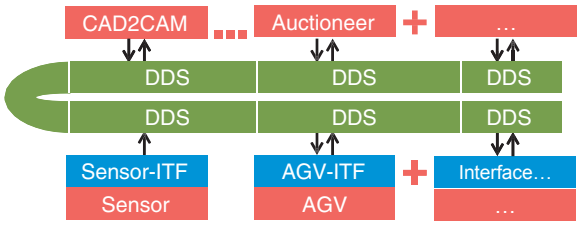

Figure 3: SInBot system proposing DDS as middleware.

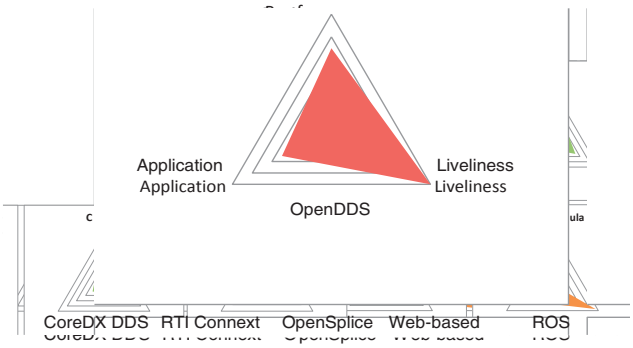

Figure 4: Performance of Communication Layer Structures.

Some software is easily integrated, while others are required to be standalone and may only use communication protocols to 'talk' to DDS entities. This results in the following proposal;

For the system to be flexible in deployment and approach the performance of high-end DDS systems, interfaces to DDS entities are created in independent scenarios: fast and (possibly) unique for demanding applications, flexible and generic for undemanding applications.

Examples of the aforementioned incompatibilities are incompatible programming languages or protected software environments. A programming language that is not natively supported by the DDS entities can be shaped into an API wrapper, e.g. 'wrapping' C++ functions in a .NET language to allow the DDS API to be controlled through the wrapper [16]. Protected software and hardware require client/server type communication, since manufacturing hardware and software often offer communication sockets / busses. An integrated approach in implementing the DDS API in manufacturing entities yields the smallest message latencies, the API wrapper will produce good results also. Navigating the messages through a client / server connection like a socket or communication bus is expected to be the least reliable and increases message latencies. The message Interactive Data Language (IDL) cannot be maintained in these cases, and is swapped for a either a 'structure > encode > connection > decode > publish' process in the case of a publisher, or a 'encode > connection > decode > structure; process in the case of a subscriber. In this example, the IDL is translated to an actual 'Structure' in the .NET framework.

As aforementioned, manufacturing environments cannot yet be expected to implement the DDS API directly into the deep programming layers of PLCs or hardware controllers. Therefore, the test software setup is based on external applications running dedicated publisher/subscriber applications. The main issue then becomes the communication of data to and from publisher or subscriber and the software. Within the Microsoft Windows environment, a variety of communication protocols between two separate applications exist: shared memory, NamedPipes, or a simple console-reading / file-reading method for instance. Each communication approach has a certain speed, complexity of implementation and compatibility with programming approaches and languages. The NamedPipe, in which a server is started to which cross-application clients can subscribe, is perceived as relatively fast, easy to implement in many different languages, easy to substitute for the UNIX-based NamedPipe definition or socket, and relatively reliable.

4.2. System Latencies

Compared to integrating the DDS API directly into the software, the NamedPipe in combination with encoding and decoding is slow. For that specific reason, it is ideal to test the performance of a DDS-based communication system in protected manufacturing (software) environments. An interpretation of this situation can be found in the SInBot system; the intelligent manufacturing entities deliberate amongst each other and the task auctioneer, which task is performed by which entity. Since these deliberations can be performed during manufacturing cycles, the message latency and frequency becomes relatively unimportant; in the order of magnitude of a few seconds, instead of micro or milliseconds.

Initial test runs showed some unreliability and increasing latencies at medium to high message frequency settings. The software was optimized for higher message frequencies, and the subsequent test runs showed enough improvement to continue this setup. The graph in Figure 6 was achieved by creating and publishing 500 messages of 0.5kb at 10 Hz (or 10 messages per second), with both the dedicated publisher and subscriber application running on a local machine, and communication event triggers for reliability (see Figure 5). The messages were created with default Quality of Service settings. In the flexible manufacturing environment as envisioned in project SInBot, Automated Guided Vehicles are required to reconfigure production cells / lines. For safety and efficiency, these AGVs rely on both internal (inertial) navigation, and information from external sensors. Solving a simple math problem; an AGV traveling at 2 m/s, assuming a required traveling accuracy of 100mm, yields a maximum latency of 50 ms.

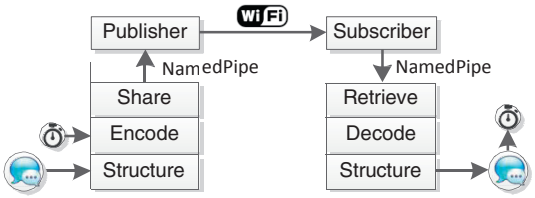

Figure 5: Communicating Manufacturing Entities.

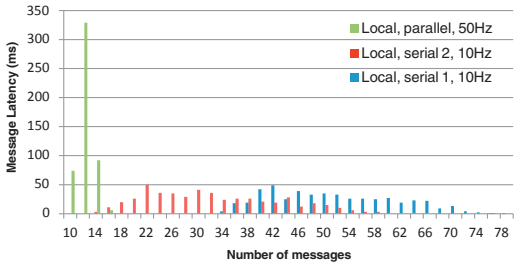

Figure 6: Difference in local message delay after optimization of serial 1 to serial 2, and parallel interpretation of values.

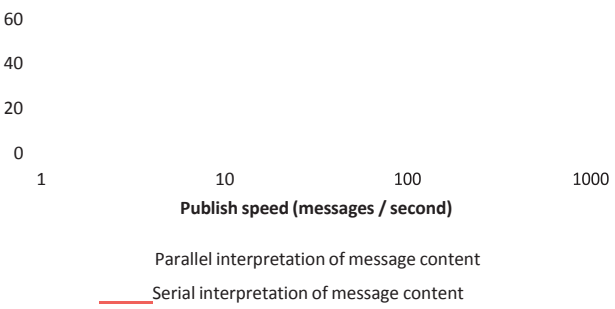

Figure 7: Reliability of communication with parallel or serial interpretation of message content in external applications.

AGVs relying solely on this information would require a direct implementation of a DDS entity in the software, as do the external positional sensors. These assumptions provide indicators to compare the performance of different interfaces to the DDS system in the form of two scenarios: 1) the flexible, non-invasive, yet relatively slow external interface, and 2) the fast, reliable, integrated approach. The serial approach of 'structurize' and 'decode', during message retrieval introduces disproportional large delays in message transport (Figure 6, serial 1). When the connection is disproportionally faster than the serial decoding process, buffer flooding occurs and reliability decreases. The key is the frequency on which the serial decoding operates. When the decoding and structurization process is placed parallel to the communication process, both the speed and consequent reliability go up (Figure 6, parallel). Compared to the initial average latency, the parallel approach averages about five times faster. Configuring the system to a speed that exceeds the serial decoding process frequency based on the outlier speed, yields message losses. The parallel approach yields perfect results with message frequencies around 70Hz, while the older configuration starts to falter above 15Hz.

It is important to note that the message is being interpreted parallel to receiving the messages, so the total delay for information has not been changed. By improving the initial latency of message retrieval from client/server communication with a dedicated subscriber, the *reliability* improved significantly. Figure 7 illustrates the effect of parallel message interpretation in terms of reliability. Next to reliability, there are some other specific advantages to this approach, especially when message structurization is not of vital importance to the receiver. Separating retrieval and interpretation enables a variable approach to interpretations, allowing some leeway to either interpret very structurally (<70Hz, clear structure of messages) or very rapid (>100Hz, only use most recent).

4.3. Conclusion

OpenDDS is considered suitable communication middleware for project SInBot. The initial latency and interface tests yield results that are within a respectable bandwidth, and within the margins expected to be required for the SInBot system to function. Embedding the DDS system is relatively straightforward, and interfacing to DDS entities can be done in multiple ways. Interpretation of communicated data has to be done parallel where possible. For a final evaluation, specific scenarios within the SInBot system are examined and build to expected specifications. Evaluating the performance of the system within these scenarios provides insight in the specific evaluation of OpenDDS in project SInBot, and shine some light of the applicability and performance in related projects and systems.

5. Specifications and Results

The proposals for communication protocols with DDS entities as mentioned in chapter 4 illustrate the possibilities for a flexible, yet slow communication scenario, and a fast but labor intensive scenario with low message latencies. Both these scenarios are common cases in the industry, where flexibility generates advantages like extendibility, ease of implementations and a work-around for protected software environments, latency-reduction enables the use of external sensor networks and safety protocols. Both scenarios are embodied in an experiment in which the latency is minimized while maximizing flexibility in deployment.

5.1. Scenario 1: Flexibility

The flexible implementation of the DDS entity consists of a computational application (.NET) that has been given the protected status. A dedicated DDS application (C++) runs simultaneously and communicates with the protected application through a NamedPipe. The information is encoded by sharing names and values (a String). The protected application uses flexible .NET libraries to distinguish what type of message has been sent through the NamedPipe, and saves the values to a structure matching the IDL. The protected application now owns a list of structures, in the case of Figure 8, a list of auctions. The DDS entities are configured to use the automated topic discovery publish/subscribe protocol, and run on a local machine, transporting messages through UDP via a Wi-Fi (wireless-n) network.

The serial message interpretation approach was done in a single thread, in which a decoder and structurize function was called following an incoming NamedPipe message. This approach yields **X~N(30,113.6)** where X is the latency of any random message, time in milliseconds. The parallel message interpretation approach was done by using an instance of a specific class running in its own thread, and starting the instance with the message coming through the NamedPipe. This approach yielded **X~N(11,1.2)**. For flexibility, the applications itself (VB.NET talker, VB.NET listener and C#.NET message monitoring application), contain only references to reusable .NET libraries (DLL). This allows for fast reproduction of .NET applications and simple connections to DDS participants. For protected applications that require DDS messages in an orderly and structured fashion but have no hard real-time requirements (maximum latency > 50 ms), this approach is ideal.

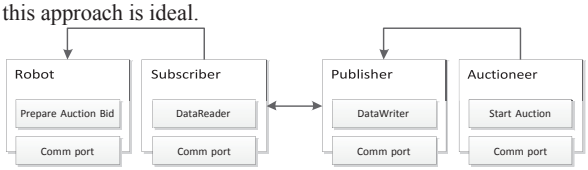

Figure 8: Example of communicating applications through NamedPipes and DDS Participants; the robot and auctioneer.

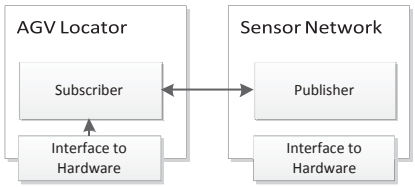

Figure 9: Example of integrated DDS participants in communicating manufacturing entities; the external sensor and AGV.

5.2. Scenario 2: Low latency

For the fast implementation, the setup contains a simple publisher and subscriber application, compiled directly from C++. Identical to scenario 1, the DDS entities are configured to use the real-time (automated topic discovery) publish/subscribe protocol, and run on a local machine, transporting messages through UDP via a Wi-Fi (wireless-n) network. This particular facet of the prototype patterns a sensor network on the publisher's side, and an application that transfers positional data to an AGV on the other side. A typical SME would have multiple individual sensor (network) entities, against a relatively small number of AGVs. The main difference as seen in Figure 9, is that the DDS entities are part of the main application, minimizing message latencies. The DDS participants are used precisely as their creator intended; to publish data when available at the available speed, and to retrieve information when needed in the required quantity. The results of this setup show an increase in speed not accurately measurable using the method from in scenario 1 (latency per message). OpenDDS published the average latency for messages being around 0.2 - 0.5 ms [17]. Even though the results show similar values, the time for the values to be ready to use by the AVG locator are somewhat longer, but remain in the <1 ms range. The time for the sensor network to produce data from perceived input is in the range of 10 ms. In this particular case, the sensor network is the bottleneck, diminishing the influence of the DDS participants on the 'age' of the data. In both the flexible and the low latency approach, running multiple subscribers on a single publisher (10:1) did not create any significant additional latencies.

Nor did adding subscriptions during the publishing process, although running the publishers and subscribers with default Quality of Service meant that only messages were received that were sent after the subscriber match. Running the same tests over a network did not yield significant changes in the results. These tests provide the first insight into the effects of a fully functional SInBot system. All local machine tests were performed on a HP EliteBook 8560w, network tests were done in combination with an Acer Aspire 5738G. Both workstations operate on Windows 7, .NET framework 4.0, OpenDDS 3.4.1, ACE 5.6 with TAO 1.6 and communicate through a wireless-n connection.

5.3. Conclusion

Building the first pieces of the SInBot system in relation to OpenDDS prove that both the flexible and fast approach are of considerable use in a manufacturing environment. Even though the message latencies are largely dependent on information generation and interpretation, rather than message transport, the fast approach can be applied when needed. Comparative transport speeds are set at 100 – 500 Hz for the fast approach, less than 100 Hz for the flexible approach, while maintaining a reliability of effectively 100%.

6. Prospects and Conclusions

Overall, the two scenarios and their software implementation perform as expected, using a local machine and validated over network. The flexible approach enables developers to prepare several applications without the need for low-level-programming or to 'hack' into protected OEM software. The reliability for DDS participants and their client/server connection to the protected applications is adequate, as is the latency of the messages. The low latencies approach performs close to OpenDDS specifications, and mean unique implementations for each manufacturing entity in a manufacturing environment; as expected. This scenario is to be used scarcely to allow true plug-and-produce systems, and only when hard-real-time requirements are encountered.

By evaluating the compatibility of DDS systems in a manufacturing environments, and their suitability for the specific problems, the production planning paradigms as stated in the problem scenario can now be properly examined. The first paradigm containing two separate robots without interaction or interference will not be a likely be a candidate for distributed and decentralized control. Paradigm 2, containing two robots that experience interference from each other, can benefit from the flexible and fast communication approach. This paradigm can be addressed by introducing a 'claimed zone' publisher and corresponding subscriber, along with a method to avoid the claimed zones. This would allow the manufacturer to program the robots without having to address possible interference. With the flexible approach, yielding a reliability close to 100%, and a message speed of up to 100Hz, end effectors can potentially work as close as 100 mm from each other, assuming an end-effector speed of ~5m/s. While this would be beneficial, the third paradigm contains even more dynamic interference; the two mobile robots that are interacting and interfering. A static work preparation approach would be exponentially more difficult than the first paradigm. Addressing such a setup could be done by no longer assigning a task to a robot, but let the robot subscribe to tasks. Next to the 'claimed zone' publisher and subscriber, a simple task scheduler could publish the next task, to which the robots can subscribe dynamically. The last paradigm introduces a human worker, involving external sensors to ensure safety. The advantage of a DDS system, is that the external sensors could simply 'claim' and publish a interference zone in which they detected an unknown object. Depending on the proximity of the object, nearby machinery could deploy a safety procedure (e.g. slow down to roughly 10% of normal speed) to ensure cooperability and safety. The required flexibility, speed, and adaptability of the communication layer is a perfect fit to the flexible, fast, and robust nature of DDS systems, as explained in the previous chapters. Finally, scaling up paradigms 2-4 increases the suitability to distributed, decentralized control systems, and re-establishes the need for the SInBot Manufacturing System.

7. References

[1] Intereg, SInBot. [cited Access 10-11 - 2013]; Available from: http://www.smartbot.eu/project-information/sinbot/.
[2] Hu, S.J., Evolving Paradigms of Manufacturing: From Mass Production to Mass Customization and Personalization. Procedia CIRP 2013; 7(0): 3-8
[3] Pardo-Castellote, G., OMG Data-Distribution Service: architectural overview. Proceedings 23rd International Conference on Distributed Computing Systems Workshops. 2003
[4] Essers, M.S. and T.H.J. Vaneker, Developing Concepts for Improved Efficiency of Robot Work Preparation. Procedia CIRP 2013; 7(0): 515-520
[5] Dianes, J.A., M. Díaz, and B. Rubio, Using standards to integrate soft real-time components into dynamic distributed architectures. Computer Standards & Interfaces 2012; 34(2): 238-262
[6] Schlesselman, J.M., G. Pardo-Castellote, and B. Farabaugh, OMG data-distribution service (DDS): architectural update. Military Communications Conference, 2004. MILCOM 2004. 2004 IEEE. 2004
[7] OMG Data Distribution Service for Real-time Systems. OMG Specification, 2007. 1.2, 260.
[8] Corsaro, A. and D.C. Schmidt, The Data Distribution Service–The Communication Middleware Fabric for Scalable and Extensible Systems-of-Systems. System of Systems 2012; 19
[9] Ryll, M. and S. Ratchev, Application of the data distribution service for flexible manufacturing automation. International Journal of Mechanical, Industrial and Aerospace Engineering 2008; 23
[10] Joshi, R., Data-Oriented Architecture: A Loosely-Coupled Real-Time SOA. Real-Time Innovations, Inc, CA, Tech. Rep 2007;
[11] Delamer, I.M. and J.L.M. Lastra, Service-Oriented Architecture for Distributed Publish/Subscribe Middleware in Electronics Production. Industrial Informatics, IEEE Transactions on 2006; 2(4): 281-294
[12] Delamer, I.M. and J.L. Martinez Lastra, Evolutionary multi-objective optimization of QoS-Aware Publish/Subscribe Middleware in electronics production. Engineering Applications of Artificial Intelligence 2006; 19(6): 593-607
[13] Guesmi, T., et al., Design and performance of DDS-based middleware for real-time control systems. IJCSNS 2007; 7(12): 188-200
[14] Woochul, K., K. Kapitanova, and S. Sang Hyuk, RDDS: A Real-Time Data Distribution Service for Cyber-Physical Systems. Industrial Informatics, IEEE Transactions on 2012; 8(2): 393-405
[15] Riquier, C., et al., DES (Data Exchange System), a publish/subscribe architecture for robotics. 1th National Conference on Control Architecture of Robots, April. 2006
[16] Calkins, C. Code Generation with OpenDDS, Part II. 2010.
[17] OpenDDS OpenDDS - Performance Testing Results. 2010.