Proceedings of the 2001 IEEE
International Conference on Robotics & Automation
Seoul, Korea - May 21-26, 2001

A High-Performance Network Infrastructure and Protocols for Distributed Automation

Shinji Kume and Alfred A. Rizzi
The Robotics Institute, Carnegie Mellon University
{shkume, arizzi}@ri.cmu.edu

Abstract

This paper documents our efforts to develop a scalable low-cost and high-performance network infrastructure for a novel distributed automation system – minifactory. *The resulting communication system (AAA-Net) relies on a suite of low-latency protocols carried by a commercial 100 Mb Fast Ethernet network. In addition to documenting the operation of this network system and its associated protocols we offer a brief comparison to the currently available commercial field network systems, and present experimental verification of the performance delivered by AAA-Net under typical and extreme operating conditions.*

1 Introduction

This paper describes a new inter-agent communication protocol which has been developed in the Microdynamic Systems Laboratory[1]. The protocol is designed specifically to serve the needs of *minifactory*, a high precision, self-calibrating, agent based, distributed assembly system which is currently under development. A minifactory system is comprised of a large collection of mechanical, computational, and algorithmically modular robotic agents which can cooperatively perform automated assembly of high-precision mechatronic devices – specifically it is targeted at hand-held and smaller devices including disk drives, PDAs, cell-phones, and the like. Minifactory is a physical instantiation of a much broader philosophy for agile assembly systems called the Architecture for Agile Assembly (AAA) [1]. The motivation behind AAA and the minifactory is to create a new standard for rapidly deployable automatic assembly systems capable of distributed execution and self-calibration [2].

Providing a suitable communication infrastructure and an associated set of protocols for seamless coordination between the physically distinct robotic agents which comprise minifactory is critical to achieving this level of capability. The inter-agent communication system described in this paper, which is termed AAA-Net, is specifically tailored to the minifactory application and uses low-latency protocols carried over a commercial 100Mb fast Ethernet network. Functionally it serves the role of an industrial field network within the minifactory system, although it is capable of significantly greater bandwidth, provides lower communication latency, and is less costly to deploy than many commercial field network systems.

[1] See `http://www.cs.cmu.edu/~msl`.

1.1 Minifactory Overview

The primary goal of AAA and minifactory is to shorten the time required to bring complex, high-precision products from a prototype phase to manufacturing production from months to weeks, or even days. Improving the responsiveness of a manufacturing system to market demands and reducing the time required to initially bring a product to market is the primary goal of agile manufacturing in general. To achieve this goal, minifactory utilizes collections of modular robotic agents, which can quickly be integrated with each other and programmed to form a highly capable, tightly coupled distributed automated assembly system [3].

Minifactory consists primarily of compact simple two degree-of-freedom robotic agents. These modular devices include an integrated mechanism, sensing systems, as well as computational and communications hardware. Furthermore every minifactory agent incorporates operational software algorithms and models making the agent truly self contained and able to describe its own capabilities to both peers and design tools.

Two basic types of agents form the backbone of the minifactory system. The first are courier robots which can move in the $x-y$ plane over the factory floor. These robots are based on planar linear motor technology [4] and are responsible for both transporting sub-assemblies through the factory and positioning them precisely while assembly operations are performed. The second class of agents are overhead manipulator robots which are capable of z and θ motions. They predominantly perform high precision sensor guided pick-and-place operations, although similar devices are also used to perform operations such as glue application, screw-driving, and the like. Undertaking assembly tasks (typically 4-DOF vertical insertion tasks) requires coordinated action by these two classes of agents. The coordination between agents can range from

0-7803-6475-9/01/$10.00© 2001 IEEE

simple semantic signaling to moderately high-bandwidth real-time cooperation, allowing two or more agents to seamlessly share their sensing and actuation capabilities.

1.2 Inter-Agent Communication

As described above, providing an appropriate communication infrastructure and framework is critical to enable reliable coordination between minifactory agents. This paper provides a brief review of existing field buss technologies, explains our rational for choosing to develop a new network system, and document that system (AAA-Net).

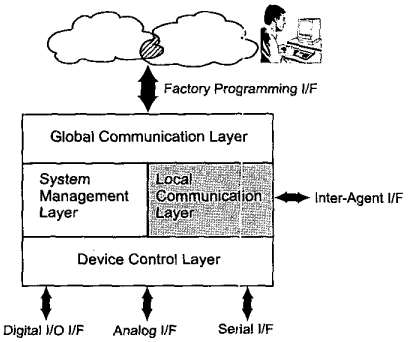

Figure 1: Software architecture of a minifactory agent.

Figure 1 shows an architectural overview of the basic software structure contained in each minifactory agent. The individual layers within this software structure are responsible for the following tasks:

Device Control Layer This layers responsible for controlling all the peripheral devices associated with the agent. Providing standard abstractions for each of the various sensors, and actuators.

System Management Layer This layer serves as the high-level commander interpreter within the agent and is responsible for the sequencing its overall operation.

Local Communication Layer This layer is the primary focus of this paper, and it is through this set of software that agents are able to perform real-time communication and coordination.

Global Communication Layer In addition to local communication, agents are also equipped with a global communication system suitable for exchanging latency insensitive information. Within minifactory communication of this class is handled via a standard TCP/IP network.

To facilitate inter-agent coordination and synchronization across the range of possible minifactory configurations, the local inter-agent communication system must satisfy a number of requirements:

Maintainability Rapid prototyping and manufacturing requires a communications infrastructure that is easy to maintain and expand.

Low Latency To support the real-time sharing of sensing and actuation resources between agents it is critical that the communications systems ensure the ability to deliver low-latency messages between controllers running on cooperating agents. For the minifactory application the target is to ensure that a cooperating pair of agents will be able to share 100 byte data packets at a rate of 1 kHz with less than 1 ms latency.

Scalability Minifactory is intended to support systems ranging in size from just a few agents, to those with up to many hundred. As such, providing a communications system that can scale to these sizes while continuing to satisfy the other requirements is necessary.

While these requirements are similar to those placed on industrial field networks under the FA open architecture system concept [5], there remain a number of notable differences. Specifically the reliance on AAA-Net to support high-bandwidth low-latency communication for executing distributed control strategies distinguishes it from the majority of its peers, which are primarily designed to support semantic communication at the cell/station level in factory. Note that minifactory actually makes use of two network systems, one for global command and control, and the second for local coordination.

2 Field Networks

The CIM (Computer Integrated Manufacturing) class model, shown in Figure 2, highlights the various levels of communication which must be supported in an automated manufacturing environment. The minifactory agents correspond to the cell/station level units (Level 2 or 3), and thus we are interested in the design of "BUS-B" in this model. "BUS-C" is also typically included in the discussion of industrial field networks, and significant effort has been dedicated to defining open standards for communication between low-level sensors and actuators, and IEEE-1451 codifies many of these ideas [6]. However, this type of communication is beyond the scope of our project as AAA and minifactory intentionally prohibits device level units from participating in such broad ranging coordination in an effort to encapsulate and localize communication. Finally "BUS-A" is responsible for transacting non real-time "global" information and and as is typical we utilize the TCP/IP protocol to support it.

2.1 Commercial Network Technologies

A variety of industrial field networks are commercially available [7, 8, 9]. Table 1 shows many of the characteris-

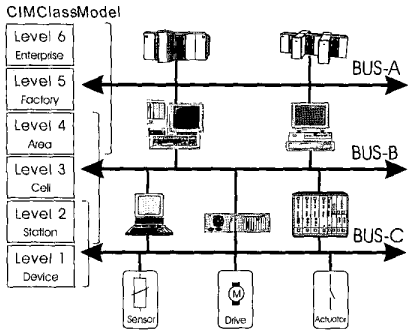

Figure 2: CIM model for communication between manufacturing entities.

	ControlNet	PROFIBUS	WorldFIP
Promoter	ControlNet Int.	Siemens Ltd.	WorldFIP
Nodes/Seg.	99	32(Max.127)	64(Max.256)
Seg. Length	250-1000m	100-1200m	500-1900m
Bandwidth	5Mbits/s	9.6-12Mbits/s	2.5Mbits/s
Topology	Bus, Star, Tree	Bus	Bus
Media Access	CTDMA	Token Passing	Bus Arbiter

Table 1: Commercial field network technologies.

tics for several of these networks.

Each industrial field network is technically distinct, and offers both advantages and disadvantages for any particular application. Our research targets the development of open architecture standards for inter-agent communication, and we are thus uninterested in closed or proprietary communications systems. Furthermore with the recent growth of demand for high-performance commercial Ethernet technology the performance and reliability offered by these commercial network technologies when compared to their cost makes them even less attractive for use in a research environment. This issues and others have driven us to consider the option of an open 100 Mb Ethernet based network system as the underlying network technology on which to base AAA-Net.

3 AAA-Net Design

3.1 Hardware

Each minifactory agent is equipped with two network interface devices. The first interface is attached to a network dedicated to transacting non-latency-critical information, such as high-level command and control messages or information destined for factory monitoring tool, and this network relies on standard IP protocols for communication. The second interface is attached to a network (AAA-Net) which is dedicated to real-time information critical for the timely coordination of activity between agents.

Unlike the majority of industrial field networks, AAA-Net makes use of commercial off-the-shelf 100 Mb Ethernet hardware. The use of such hardware components has given us access to a wide variety of low-cost and compact hardware devices, reducing our investment in network infrastructure hardware as well as minimizing instillation and maintenance costs.

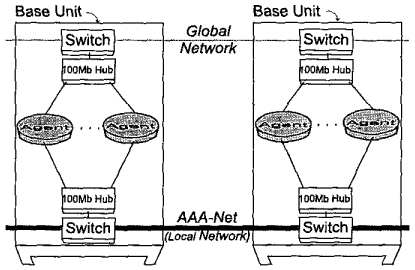

Figure 3: Network topology of the minifactory system.

Figure 3 depicts the basic topology of the communications infrastructure used in minifactory. As can be seen, both the AAA-Net and the global IP network are configured as a chain of star topology local-networks, with a Fast Ethernet hub at the center of each star and Fast Ethernet switches forming the connections between the local-network segments. The hub allows each agent in a network segment to directly communicate with its immediate neighbors, while the frame relay switches allow essential arbitrary daisy chaining of the network, allowing the system to grow easily. Scalability is ensured through the use of the switches, as they serve to localize communications within the factory system by not transmitting data packets destined for local agents to the remainder of the factory.

3.2 OSI Reference Model and AAA-Net

Most data communications protocols in use today are defined as a layered model called the Open Systems Interconnection (OSI) Reference Model [10]. Thus, we digress briefly to describe AAA-Net by reference to the OSI Reference Model in order to make the relationship clear.

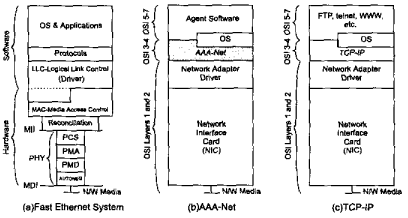

Figure 4: Relationship between OSI reference model and AAA-Net.

Figure 4 (a) shows the hardware components in a Fast Ethernet network together with a simplified depiction of software layers that reside above the hardware. The AAA-Net, which resides at the protocol layer, is responsible for translating agent software data at the application layer into a appropriate format for transmission over the physical

network (see Figure 4 (b)). Figure 4 (c) depicts the non-latency-critical network based on the TCP/IP protocol.

3.3 Performance Evaluation

In a Fast Ethernet network, the media access rule is a CSMA/CD (Carrier Sense Multiple Access with Collision Detection) policy. The CSMA/CD media access rule is defined in ANSI/IEEE Std 802.3 and is also used in standard Ethernet networks [11]. This policy has a significant influence on the data transmission performance of the network system, especially when the network system is "busy." Unfortunately, due to the structure of the CSMA/CD rule, the timing of packet delivery is not deterministic. We are thus forced to consider the range of possible back-off times (described below) in order to validate the use of Fast Ethernet as the underlying infrastructure for delivering latency-critical information via AAA-Net.

The CSMA/CD media access rule accepts the occurrence of collisions. If multiple nodes (multiple agents in our case) attempt to send data simultaneously onto the Fast Ethernet, the hub in that network segment detects the collision and immediately sends a 32-bit long JAM pattern out to all ports. The transmitting nodes detect this signal and the MAC (Media Access Controller) increments an internal counter (transmit attempts). The node then attempts to retransmit the packet after waiting a random period of time – this delay is the "back-off" time.

The back-off time generated by the MAC is an integer multiple of slot times, where one slot time is equal to 512 bit times and one bit time is 0.01 μ-seconds for Fast Ethernet. The number of slot times to delay is chosen as a uniformly distributed random integer in a range that grows with retransmission attempts.

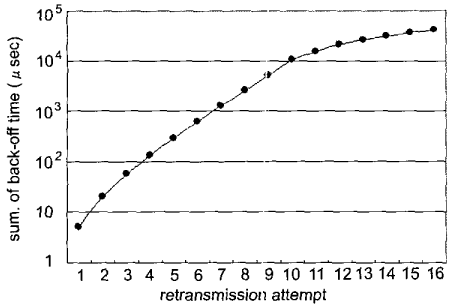

Figure 5: Worst case back-off time vs. retransmission attempt.

Figure 5 shows the maximum possible delay from the initial attempt to transmit a packet for each retry. From this figure we see that we can ensure packet delivery in less that 1 ms if fewer than 6 retransmission attempts are required. Based on the maximum expected loading of AAA-Net (8 agents each transmitting 100 byte data packets at a rate of 1 kHz) the probability of a packet requiring six retransmissions is approximately one in two-million, or roughly once for every 30 minutes of full network utilization. The probability of delaying a packet is small enough for us to conclude that the Fast Ethernet media access rule will not significantly impact overall system performance.

3.4 AAA-Net Services

The AAA-Net software has been developed in accordance with POSIX.4, resulting in software that is maximally portable. To facilitate a wide variety of local interactions between agents, AAA-Net provides two basic network protocols. The first is a connection-less non-guaranteed transmission protocol which seeks to provide network performance. The second is a connection based guaranteed transmission protocol which provides for reliable communications. These services are not unlike the familiar UDP and TCP services commonly used on IP networks, although they are specifically tailored to improve performance in the highly structured network environment of minifactory. The two protocols provided exhibit significantly different characteristics as summarized in Table 2.

	Non-Guaranteed Data Transmission	Guaranteed Data Transmission
Connection-based	No	Yes
Error Control	No	Yes
Flow Control	No	Yes
Time-Out & Retransmission	No	Yes
Sequencing	No	Yes

Table 2: Comparison of protocol features in AAA-Net.

The AAA-Net software spawns a server process on the agent's computer, and this server interacts with the remainder of the agent software to provide the needed communication facilities. Agent software communicates with the AAA-Net server through a collection of library routines and is presented with a flexible abstract view of the underlying communication system, while the AAA-Net server takes responsibility for processing those requests, formatting and transmitting raw-Ethernet packets to its peers, and processing incoming AAA-Net packets.

3.4.1 Non-Guaranteed Data Transmission

Internally AAA-Net makes use of two threads to handle non-guaranteed communication. The first thread is responsible for processing internal requests from the agent software — including requests to listen for messages on a particular port, and requests to send messages to peers. The second listens for packets arriving over the network and routes them to the appropriate location internally in the agent. Figure 6 shows a simplified overview of the frame format used by this data transmission protocol.

Figure 6(a) depicts the standard frame structure used by Fast Ethernet. The minimum length of the data field is 46 bytes and the maximum length of it is 1500 bytes.

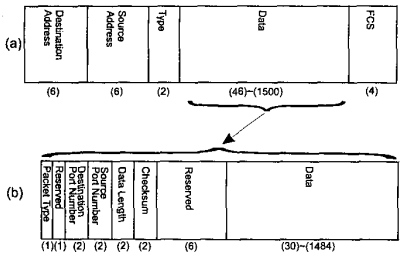

Figure 6: Non-guaranteed AAA-Net packet format.

The TYPE field is the "Ether Type" of the packet, and is defined by RFC 1700. The FCS (Frame Check Sequence) is used to ensure that the frame has been received without errors by the destination agent. Figure 6(b) shows the the use of the DATA field for a non-guaranteed data transmission. The PACKET TYPE is set to 0 to identify that this packet is a AAA-Net non-guaranteed data transmission. The DESTINATION PORT NUMBER field and the SOURCE PORT NUMBER field are used to specified by the agent software to indicate routing of the packet on the remote agent.

Once formatted, the packet is then placed directly on the Fast Ethernet network with no effort made to guarantee its delivery. This form of communication is appropriate for the exchange of rapidly changing values, such as the velocity commands sent from one agent to the other, where the loss or delay of a single packet is insignificant since additional data will follow shortly.

3.4.2 Guaranteed Data Transmission

To create and maintain a reliable connection between agents, the AAA-Net process must maintain internal state describing the condition of a connection. This state information is maintained by a temporary protocol management thread which has essentially three states:

Start: Waiting to establish a communication.

Ready: The data transmission state.

Finish: Waiting to close a connection.

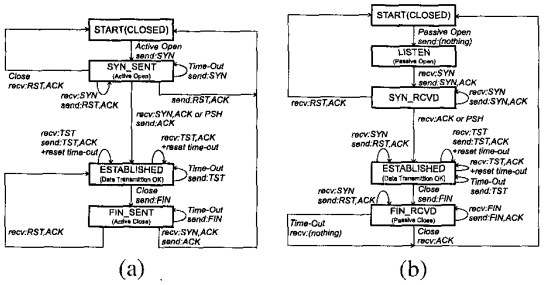

Figure 7: State transition diagram for AAA-Net guaranteed delivery protocol: (a) "client", (b) "server".

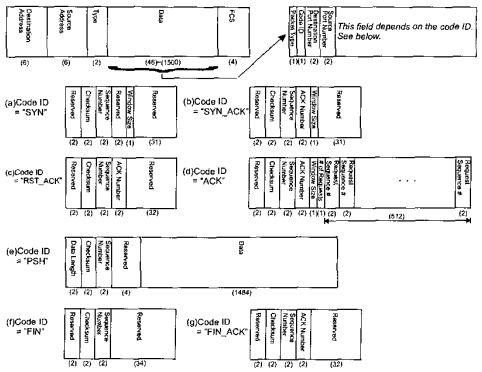

Figure 8: Guaranteed delivery AAA-Net packet formats.

When starting the connection one agent (referred to as the *server*) waits for its peer (the *client*) to initiate contact. The two then exchange a small number of packets (executing a three-way handshake) to acknowledge the creation of the connection. Once the connection is established the use of a sliding window scheme and acknowledge messages ensures that data is delivered in sequence until the connection either fails or is terminated. To facilitate the detection of network difficulties, each agent transmits a small TEST packet periodically to verify the integrity of the network. Figure 7 depicts the state transition diagram used by this protocol.

The Fast Ethernet packet format used by the guaranteed delivery protocol is shown in Figure 8. Here, the PACKET TYPE is set to 1, and the added fields following the SRC. PORT are used to carry additional information needed by the protocol to ensure reliable delivery of the data stream. Figures 8(a)-(g) show the format of the various packets exchanged by the AAA-Net software to implement this protocol.

4 Experimental Evaluation

4.1 Non-Guaranteed Delivery Protocol

A number of experiments were performed to evaluate the performance of the non-guaranteed AAA-Net protocol and underlying network infrastructure. First, two agents were connected to one another via a hub. To distinguish the performance of the network itself and the associated software overhead, timing data was collected at several points both in the AAA-Net process and the application-level process. Those points are shown in Figure 9, here only the sender's processes are depicted but an identical AAA-Net process and application-level processes are receiving and sending back data on the peer agent.

Figure 10 shows the timing data recorded at each of these test points. A data payload of 100 bytes was used to generate these data. From these data we quickly conclude

	Round-trip (μs)							
Number of	100 byte message				1000 byte message			
Agents	mean	std	max	min	mean	std	max	min
2	566	20	2074	544	804	22	2369	775
4	562	19	1971	539	816	34	2573	779
6	567	21	2139	547	826	38	2218	779

Table 3: Round trip transmission times for various levels of network loading.

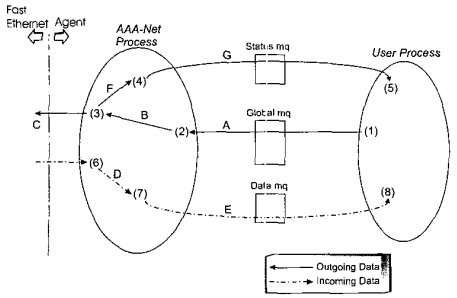

Figure 9: Instrumented locations for evaluating AAA-Net performance.

that the total transmission time between application-level processes on two agents will be less than 300 μs under ideal conditions.

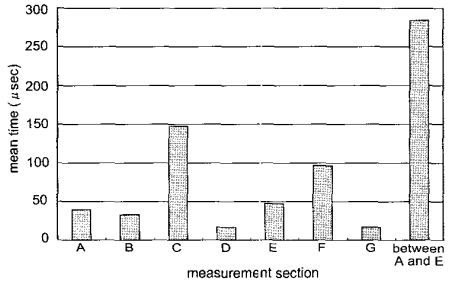

Figure 10: Timing results for non-guaranteed protocol.

Next, a collection of agents were connected to the same network segment to simulate a more realistic operating condition. Messages were exchanged between all the agents at 1kHz, with data sent from one agent, received by another agent, and retransmitted back to the original agent. Overall round-trip time was measured for each message. To evaluate the impact of network loading this test was performed with one, two, and three pairs of agents communicating simultaneously. Messages of both 100 and 1000 bytes in length were transmitted, and the results of 30 seconds of operation are shown in Table 3.

From these data we conclude that the relatively low loading we place on the Fast Ethernet (even at at loads far in excess of worst case minifactory loading) allows us to ensure timely delivery of information even over the non-deterministic Fast Ethernet infrastructure.

Acknowledgments

This work was supported in part by NSF grants DMI-9523156 and CDA-9503992, and by the Mitsubishi Materials Corporation . The authors would like to thank Ralph Hollis, Arthur Quaid, Zack Butler, Jay Gowdy, and Michael Chen for their invaluable work on the project and support for this paper.

References

[1] R. L. Hollis and A. Quaid. An architecture for agile assembly. In *Proc. Am. Soc. of Precision Engineering, 10th Annual Mtg.*, Austin, TX, October 15-19 1995.

[2] A. A. Rizzi, J. Gowdy, and R. L. Hollis. Agile assembly architecture: An agent-based approach to modular precision assembly systems. In *IEEE Int'l. Conf. on Robotics and Automation*, pages Vol. 2, p. 1511–1516, Albuquerque, April 1997.

[3] A. A. Rizzi and R. L. Hollis. Opportunities for increased intelligence and autonomy in robotic systems for manufacturing. In 8^{th} *International Symposium of Robotics Research*, Hayama, Japan, October 1997.

[4] B. A. Sawyer. Multi axes linear movement positioning system. U.S. Patent 3,836,835, September 17 1974.

[5] Yoshikawa and Narusawa. Open system architecture for servo drivers in servo systems division. Technical Report 3, Sanyo Denki, 1997.

[6] K. B. Lee and R. D. Schneeman. Distributed measurement and control based on the IEEE 1451 smart transducer interface standards. *IEEE Transactions on Instrumentation and Measurement*, 49(3):621–7, June 2000.

[7] Profibus Home Page. http://www.profibus.com.

[8] ControlNet Online. http://www.controlnet.org.

[9] WorldFIP Home Page. http://www.worldfip.org.

[10] U. Black. *OSI: A Model of Computer Communications Standards*. Prentice Hall, Englewood Cliffs, N.J., 1991.

[11] IEEE 802.3u-1995 10 & 100 Mb/s Management, October 1995. Section 30, Supplement to IEEE Std 802.3.