Proceedings of the 1994 Winter Simulation Conference
ed. J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila

AN AGENT-BASED FLEXIBLE ROUTING MANUFACTURING CONTROL SIMULATION SYSTEM

Grace Y. Lin

IBM T.J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598, U.S.A.

James J. Solberg

School of Industrial Engineering,
Purdue University
West Lafayette, IN 47905, U.S.A.

ABSTRACT

In this paper, we present a manufacturing simulation system based on autonomous agents and a dynamic price mechanism that explores routing flexibility and provides a programming language for modeling manufacturing environments, based on autonomous agents and a dynamic price mechanism. The simulation system contains a control simulation module and a manufacturing environment simulation module. The control simulation module consists of a collection of autonomous agents who negotiate with each other to reach job processing decisions based on negotiation protocols and built-in price adjustment algorithms. The manufacturing environment simulation module is an event-based simulation system that couples with an input simulation language called Flexible Routing Adaptive Control Simulation Language. The language can be used to model complicated manufacturing environments and to specify flexibility in part process plans. By integrating the control framework with the FRACS simulation system, a sophisticated test bed is created for research of different control and negotiation strategies. The simulation system allows quick turn around time for software prototyping, and the modeling language enables easy model adjustment and performance tuning. Many experiments in the control and scheduling of manufacturing systems have been conducted using this simulation system. Several of these experiments are discussed in this paper.

Key Words: Manufacturing Control, Scheduling, Simulation, Flexible Routing.

1 INTRODUCTION

Modern manufacturing environment is complex and uncertain. In order to stay competitive in the market place, the decision-making tools need to have the ability to adapt to the changing environment and to handle the system complexity. The ability to make use of the system flexibility and to retain flexibility itself is a primary requirement for the decision tools to be effective. To meet this requirement, we proposed an agent-based real time adaptive control and scheduling framework [Lin and Solberg, 1992]. Under the framework, both decision making and information flow are distributed so that it is flexible and can adapt quickly to the changes. All possible sequencing and processing of a job are considered in real time during the decision making process to make use of the system routing flexibility and to improve system throughput. The global and local system states are captured using a price mechanism. The price mechanism is used as an invisible hand to guide the negotiation and ensure harmonious system operation.

Due to the complexity of the manufacturing environment, it is very expensive and time consuming to setup a manufacturing floor to do experimental testing and comparison of different control strategies, negotiation protocols, or factory setups. On the other hand, software prototyping by simulation offer a quick, controllable, tunable environment for performance the above tasks before applying the control system to real manufacturing environments. The software prototyping system must offers a good modeling tool for modeling the manufacturing environments. Important system entities need to be included in the model so that the modeling is realistic. It also needs to provide flexible interface so that changing control strategies, negotiation protocols, and factory setup can be accomplished easily. Furthermore, our framework requires that unexpected events, such as machine breakdowns, order changes, supply problems, and objective changes, be modeled in the simulation. None of the existing simulation systems as we noted provide these functionalities. In this paper, we will present an integrated shop floor control and simulation system that models the proposed framework. The simulation system supports the modeling of system resources such as machines, automatic guided vehicles (AGVs), tools, buffers, etc., and part process plans that contain alternative processing sequences or operations. The system resources and parts can be controlled by simple dispatching rules or autonomous agents that employ sophisticated control protocols. The system supports an extensive set of debugging and trace facilities and visualization tools for analyzing experimental results. The system is carefully designed so that by decoupling the control and scheduling system from the simulation module and linking the former to the real time control and monitoring interface of a manufacturing environment, a real time manufacturing control system can be created. This is accomplished by separating the simulation system into two modules, a control simulation module and a manufacturing environment simulation module, and integrating the two simulation modules with three interface modules: the control interface, the monitoring interface, and the communication interface.

By using the built-in negotiation protocols, we were able to implement new negotiation algorithms in days rather than months. Our experiences in applying the simulation system show that software prototyping can shorten development cycles and provide quick turn around time for system design and changes.

The rest of this paper is organized as follows: in Section 2, we will briefly describe the flexible routing control framework. The detailed simulation system will be presented in Section 3. In Section 4, we will describe some experiments performed using the simulation system. A summary will be given in Section 5.

2 AN AGENT BASED REAL TIME CONTROL AND SCHEDULING FRAMEWORK

Under our framework, the control and scheduling system is modeled as a marketplace. Each system entity is equipped with a software agent who controls the functioning of the entity and represents the entity to negotiate with other agents. A part equipped with a part agent enters the system with some (fictional) currency, an objective function, and *a flexible processing process plan* (see [Lin and Solberg, 1991]). It tries to fulfill its processing requirement to achieve its objective by bargaining with resource agents. Each resource agent labels its processing charge according to its status and tries to sell its service to maximize its profit. That is, the manufacturing system is considered as a collection of intelligent system entities. The entities negotiate with each other by message passing for task sharing to achieve their individual goals. Mutual selection and mutual agreement are

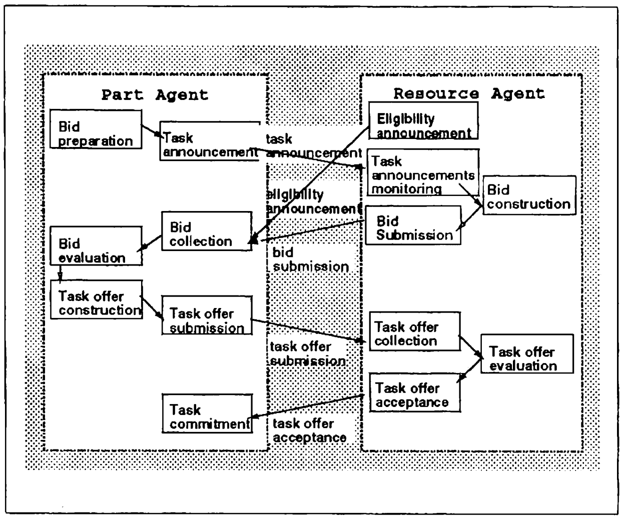


Figure 1: A general negotiation protocol

made through multiple-way communication. The system functions based on the equilibrium of the price system and the part's objective. The high currency part will be able to achieve a better objective value and the critical resource which may change over time will play a major role in decision making. Therefore, the decision making process is called a dynamic critical resource centered decision making scheme.

The general negotiation protocol is depicted in Figure 1. Under the protocol, both parts and resources have the capability to initiate the negotiation procedure. They will then collect information from the network and select the entities to submit their constructed bids using their own evaluation schemes. The negotiation procedure is completed when both part and resources commit to a deal.

The agent-based real time control and scheduling framework has several advantages over conventional control and scheduling systems. First, global states are usually unavailable in real manufacturing environments; this makes centralized decision making unrealistic. The dynamic price mechanism captures the system status and reflects the priority and bottlenecks in price fluctuation. The distributed decision making by cooperation among agents can utilize local states and the price mechanism to improve system performance by improving performance of each individual entity. Second, the agent-based negotiation framework is robust and can cope with sudden environment changes. Third, different entities can have different objectives. The behavior of the agents can be controlled by changing objectives, price adjustment algorithms, audience selection, bid construction methods, etc. Fourth, the framework supports "plug and play" different negotiation protocols, and price adjustment algorithms can be used. Deriving new negotiation algorithms based on built-in generic negotiation protocols is fairly straight forward.

3 AN AGENT BASED SIMULATION SYSTEM

We designed an agent based control simulation system to model the agent based real time control and scheduling framework. The system contains a control simulation module and a manufacturing environment simulation module. The control simulation module consists of a collection of autonomous agents who negotiate with each other to reach job processing decisions based on the negotiation protocols and the built-in price adjustment algorithms. The manufacturing environment simulation module is an event-based simulation system that couples with an input simulation programming language called Flexible Routing Adaptive Control Simulation Language. The language has provisions to model components of manufacturing environments such as machines, AGVs, tools, tool storage units, stamping stations, central buffers, etc. It also allows the part process designer to express flexibility in part process plans. The integration of the two modules is based on three interface modules: the control interface, the monitoring interface, and the communication interface.

The control interface provides a set of functions for controlling the activities of part entities and resources in the manufacturing simulation system. The monitoring interface module supports a set of queries for obtaining states of part entities, resources, or the global states of the environment.

By integrating the control framework with the FRACS simulation system, we created a sophisticated testbed for experimenting with different control and negotiation strategies. By decoupling the control and scheduling system from the simulation system and linking it to the real time control and monitoring interface of a manufacturing environment, a real time manufacturing control system can be created. This arrangement allows us to test the proposed concepts in the controlled simulation system and also allows the control system to be easily ported to real time control of manufacturing systems.

All communication between agents in the controlled system or between the control system and the controlled units and sensors goes through a communication interface module which is an abstraction of the underlying communication network and provides utilities for sending and receiving messages, broadcasting messages, and transmitting commands and bids, etc.

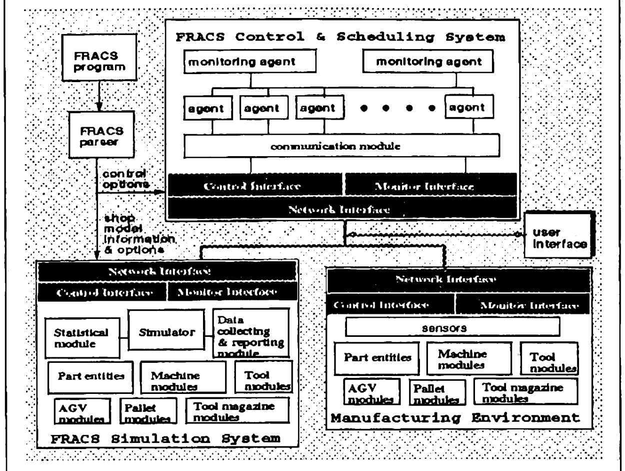


Figure 2: The agent based control simulation system

The simulation system and the control system are both event driven. The simulation system is driven by the time-dependent events generated by the simulator. And the control system is driven by the events caused by receiving messages from other agents or from interface modules. The modular system description is shown in Figure 2.

In the following section, we will first describe the manufacturing environment simulation system and the control simulation system. The monitoring, control, and the communication interface modules will then be discussed.

3.1 The Manufacturing Environment Simulation System

The manufacturing environment simulation system contains a very high level programming language called the FRACS programming language, a parser for the FRACS language, an event-driven simulator, a data and statistics collection system, a trace facility, a visualization interface, and an interface to the control system (Figure 3).

Simulation Control

The functioning of the simulation system can be controlled through the input FRACS program or command line options. The following global simulator variables can be set in the FRACS program:

buffer_size - default size of central buffer.
run - run the simulator for this many times.
seed - seed for the random number generator.
rule - negotiation methods or dispatching rules.

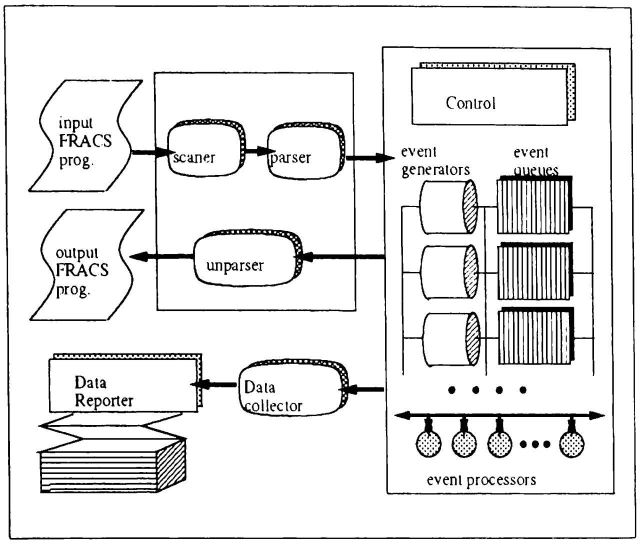


Figure 3: The manufacturing environment simulation system

trace - time period for generating trace information.
in_buffer_size - default machine in buffer size.
out_buffer_size - default machine out buffer size.
report_period - time period between intermediate statistical results are generated.

Other control variables for the simulated manufacturing system are set based on the resource and part specifications in the FRACS program. FRACS programs specify the factory setup, capabilities of the resources, part process plans, and the control and simulation options.

Input FRACS Program

The very high level programming language approach of FRACS provides a flexible and easy way for the user to model and define the manufacturing environments. The FRACS language contains constructs to model manufacturing environments, part process networks, and the control structure and strategies. The modeling of the environment includes the definition of machines, transporters, central and local buffers, machine breakdowns, factory layout geometry, initial system states and prices, and price adjustment parameters. Parts description, objective function, arrival rate, and price setting are also specified. The part process network is represented by operation nodes and the precedence arcs. The operation nodes contain the desired operations, constraints of the operations, and the types of precedence arcs originating from the nodes. The precedence arcs determine the execution orders of the operation nodes. The specifications of the control include the control algorithm and strategies, dispatching rules, evaluation functions, and predefined optimization options. in Appendix.

Simulator Logic

The logic of the simulator is as follows: new part entities arrive at the system at a rate defined by the arrival functions specified in the FRACS program. At any instant, the state of a part entity inside the system is one of the following: being served by a machine, waiting to be served in the input buffer of a machine, waiting for transportation in the central buffer or output buffer of a machine, or being moved. A new part entity or a part entity whose operation is just finished by a machine sends a status report to the control simulation module to determine its action; it will either initiate a negotiation process or collect resource information and participate in an existing resource initiated negotiation process depending on the control module. Part and machine controls then negotiate with each other according to their control protocol and evaluation functions to match a ready operation (or several operations if look ahead control is invoked) of a part entity to a machine taking consideration of the transporter availability. Once the desired operation and machine matches are done, if the required immediate service machine has a space to hold the part entity, the part entity requests a transporter and waits to be moved to the input buffer of the designated machine. By default, part entities in the local input buffer of a machine get service based on the first-come-first-served dispatching rule. When a part is served, an end-of-service event is scheduled to simulate the completion of the operation. Upon completion of the operation, the part entity requests a transporter to move it elsewhere unless the next operation will be processed by the same machine. If the machine has space in the output buffer, the part entity whose operation was just finished will be moved to the local output buffer of the machine; otherwise, the machine is blocked by this part entity and cannot resume operation until the part entity is moved. When the transporter arrives, if the part entity has not found a machine or the assigned machine has no place to hold it, it will be temporarily moved to the central buffer to free up the output buffer of the machine to avoid blocking. This process continues until the part entity is finished processing. Breakdown events are scheduled according to the up-time functions of the machines specified in the FRACS program. When a machine is down, parts that were assigned to the machine (being processed by the machine, in the local input buffer of the machine, being moved to the machine by a transporter, or in the central buffer or local output buffer of other machines waiting for transporter assignment or arrival) are rescheduled and moved to a new destination machine or central buffer (if no machine is available). And a fixed event is scheduled according to the down-time function specification. When a machine is up again, it resumes its operation by declaring its status to be idle and invokes the machine control routine to look for parts.

Command Line Options

The simulation system also provides command line options to overwrite or ignore some of the options defined in the FRACS program for convenience in conducting experiments. For examples, -i flag allows the user to input the part program interactively, -C flag allows the user to specify the central buffer size. Other options include the specification of look ahead time, bidding strategies, number of operations to announce at a time, and the output format.

System Output

The output of the system includes the trace information, debugging information, statistical information, and final results. All the output can be controlled with an extensive set of options.

Trace Information

Trace information can be generated for monitoring and understanding the control and simulation systems. The simulation system generates a human readable trace for understanding the behavior of the system. The example listed in Figure 4 shows a trace file produced by the FRACS system during the specified ten minute period. The trace data provides detailed information about selected events in the system.

The system also generates a different status and statistical trace that can be fed into an X-window visualization package to give a detailed graphical view of the progress of the system. Figure 5 contains an X-Window dump of some graphical displays showing the statistical data. Currently, this tool shows the graphs in postmortem mode only.

Debugging Information

The system has a built-in debugger so that it can be used to trace system problems or improve setups in a user's program. A flag can be set so that the system does a sanity check periodically to uncover inconsistency in the system. The system will output debugging information when the debugging flag is set to a positive number (that is less than 10). The larger the value of the flag, the more output will be produced.

As with all debugging processes, using the debugger is an involved job. The granularity of the debugging information is based on functions and is much finer than the trace facility. Typically, the system

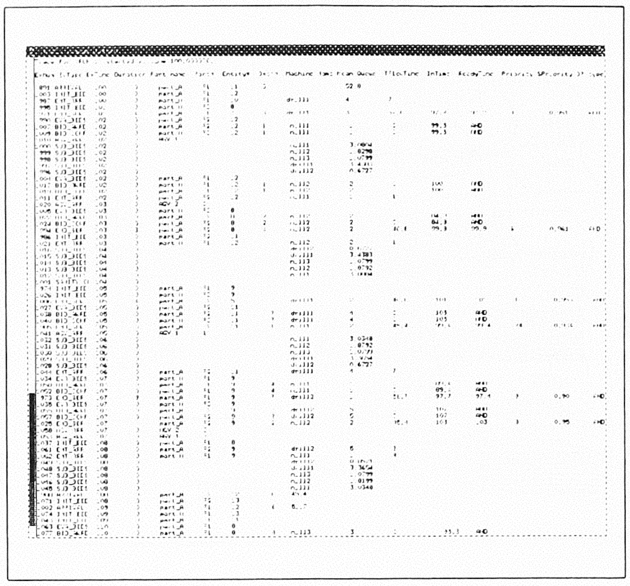


Figure 4: The trace output window

generates multiple megabytes of debugging information for small programs. The user should start with a small setting of the debugging flag, narrow the problem to a small time interval, and then set the debugging flag to generate detailed information to actually find the problem.

In Figure 6, we list a fragment of the debugging information that was produced with the debugging flag set at 2. get debug output from file ndebug.out

Statistics and Results

The result of simulation is summarized in a summary report that includes a list of options and summarized information about parts and resources such as part mean flow time, throughput, price, objective value, machine price, utilization, and maximal and minimal central queue size. Figure 7 shows a fragment of a typical report that is generated.

3.2 A Negotiation-Based Adaptive Control System

The negotiation-based adaptive control system is composed of a set of autonomous agents who negotiate with each other to direct the activities in the manufacturing environment. A part-resource negotiation process begins when a new part equipped with a part agent enters a loading station, a task of the part entity is near completion, or when the previous negotiation process fails to yield any binding bids. Agents in the control module are equipped with event handlers. Each can invoke needed system knowledge such as price adjustment scheme, bid

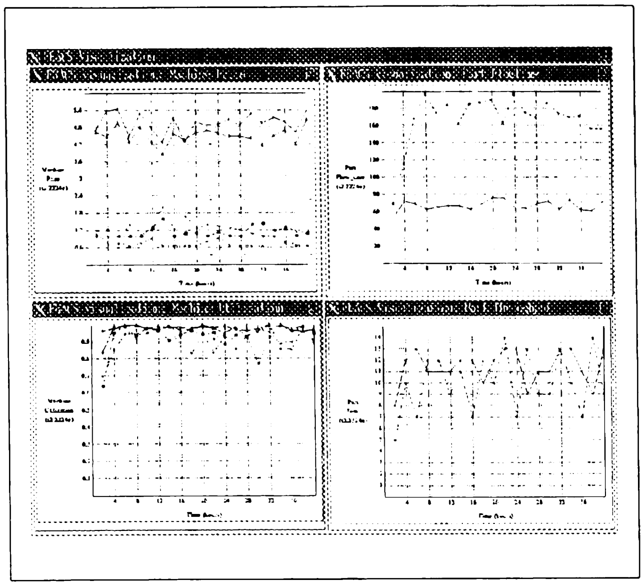


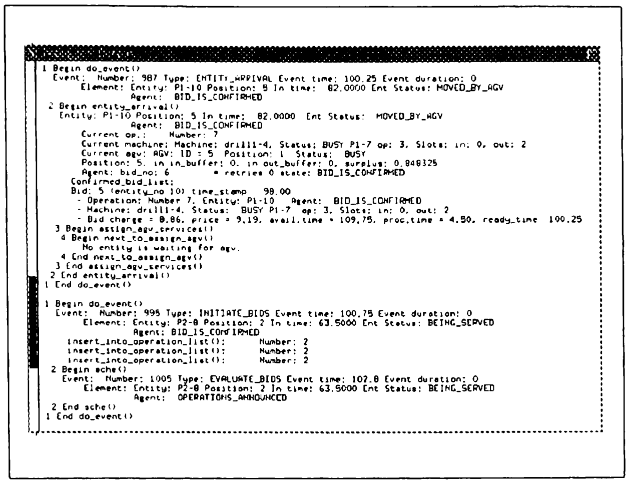
Figure 5: The window dumps of the graphic visualization of trace data

construction, and submission for decision making. Each event handler is supported by a timer and an event queue. The event handler paces the agents and decides when to generate and handle certain events. Examples of the events include INITIATE-BIDS, EVALUATE-BIDS, BID-CONFIRMATION-TIMEOUT, RE-NEGOTIATION, etc. Options are provided in FRACS language to control the scheduling of these events.

The system currently supports the following different multi-stage negotiation protocols. See [Lin, 1993] for detailed descriptions.

- Part-initiated, resource-centered negotiation protocol.
- Part-initiated, resource-centered negotiation protocol with multiple bid ahead.
- Resource-initiated, part-centered negotiation protocol.
- Resource-initiated, part-centered negotiation protocol with multiple bid ahead.
- Resource-initiated, bottleneck-centered negotiation protocol.
- Resource-initiated, bottleneck-centered negotiation protocol with multiple bid ahead.

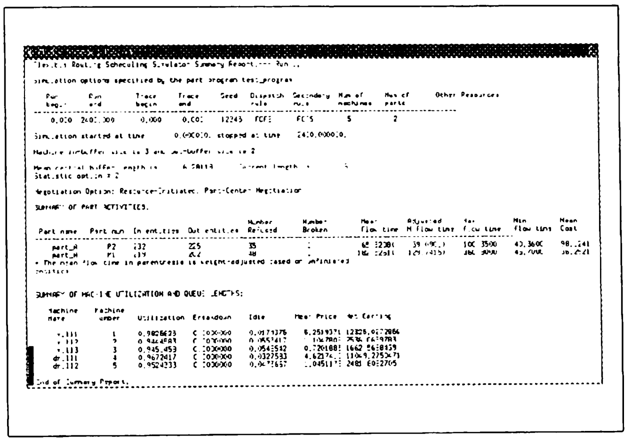
The current protocol is set by the variable control_method and the number of bids to bid ahead is



```
1 Begin do_event()
  Event:  Number: 987 Type: ENTITY_ARRIVAL Event time: 100.25 Event duration: 0
       Element: Entity: P1-10 Position: 5 In time:  82.0000 Ent Status: MOVED_BY_AGV
                Agent:  BID_IS_CONFIRMED
 2 Begin entity_arrival()
   Entity: P1-10 Position: 5 In time:  82.0000  Ent Status:  MOVED_BY_AGV
                Agent:  BID_IS_CONFIRMED
       Current op.:    Number: 7
       Current machine: Machine: drill1-4, Status: BUSY P1-7 op: 3, Slots: in: 0, out: 2
       Current agv: AGV: ID = 5  Position: 1  Status:  BUSY
       Position: 5, in in_buffer: 0, in out_buffer: 0, surplus: 0.848325
       Agent: bid_no: 6        * retries 0 state: BID_IS_CONFIRMED
      Confirmed_bid_list:
      Bid: 5 (entity_no 10) time_stamp   98.00
       - Operation: Number 7, Entity: P1-10   Agent:  BID_IS_CONFIRMED
       - Machine: drill1-4, Status:  BUSY P1-7  op: 3, Slots: in: 0, out: 2
       - Bid charge = 8.86, price = 9.19, avail.time = 109.75, proc.time = 4.50, ready_time  100.25
  3 Begin assign_agv_services()
   4 Begin next_to_assign_agv()
       No entity is waiting for agv.
   4 End next_to_assign_agv()
  3 End assign_agv_services()
 2 End entity_arrival()
1 End do_event()

1 Begin do_event()
  Event:  Number: 995 Type: INITIATE_BIDS Event time: 100.75 Event duration: 0
       Element: Entity: P2-8 Position: 2 In time: 63.5000 Ent Status: BEING_SERVED
                Agent: BID_IS_CONFIRMED
     insert_into_operation_list():      Number: 2
     insert_into_operation_list():      Number: 2
     insert_into_operation_list():      Number: 2
 2 Begin sche()
   Event:  Number: 1005 Type: EVALUATE_BIDS Event time: 102.8 Event duration: 0
       Element: Entity: P2-8 Position: 2 In time: 63.5000 Ent Status: BEING_SERVED
                Agent:  OPERATIONS_ANNOUNCED
 2 End sche()
1 End do_event()
```

Figure 6: Sample debugging information with debugging flag set at 2



```
Flexible Routing Scheduling Simulator Summary Report.--- Run ..

Simulation options specified by the part program test_program

 Run      Run       Trace    Trace     Seed   Dispatch  Secondary  Num of    Num of      Other Resources
 begin    end       begin    end              rule      rule       machines  parts
----------------------------------------------------------------------------------------------
  0.000  2400.000   0.000    0.000    12243   FCFS      FCFS        5         2

Simulation started at time      0.000000, stopped at time    2400.000000.

Machine in-buffer size is 3 and out-buffer size is 2

Mean central buffer length is     6.[illegible]     Current length is      5
Statistic option = 2

Negotiation Option: Resource-Initiated, Part-Center Negotiation

SUMMARY OF PART ACTIVITIES.

                                                 Number     Number      Mean        Adjusted     Max        Min        Mean
 Part name  Part num  In entities  Out entities  Refused    Broken      flow time   M flow time  flow time  flow time  Cost
-----------------------------------------------------------------------------------------------------------
   part_A      P2      232          225           35          [illegible]  62.32300   39.[illegible]  100.3500   40.3600   98.[illegible]
   part_B      P1      [illegible]  [illegible]   48          [illegible]  [illegible] [illegible]  [illegible] [illegible] [illegible]
 * The mean flow time in parenthesis is weight-adjusted based on unfinished
 entities

SUMMARY OF MACHINE UTILIZATION AND QUEUE LENGTHS:

  Machine   Machine
  name      number   Utilization  Breakdown    Idle        Mean Price  Net Earning

   [illegible].111    1     0.9826623    0.0000000    0.0173376    8.2519371  12326.0272866
   [illegible].112    2     0.9424583    0.0000000    0.0557417    [illegible]  2536.6619783
   [illegible].113    3     0.945[illegible]453    0.0000000    0.0543542    0.7201883  1662.5638439
   dr.111       4     0.9672417    0.0000000    0.0327583    4.62174[illegible]  1069.2753471
   dr.112       5     0.9524233    0.0000000    0.04[illegible]7657    [illegible].0451173  2481.6032705
End of Summary Report.
```

Figure 7: Sample summary report for the simulation results

determined by the variable number_bid_ahead. Our experiments show that excessive bid ahead can cause agents of parts and resources to make premature commitments and hurt the performance. A one or two bid ahead is recommended for most cases.

The framework of the system also provides hooks for using different algorithms for price adjustments and resource selections. Several different price adjustment strategies and methodologies for selecting resources or selecting parts to serve are also supported. Each of the price adjustment and selecting algorithms can be used by each of the negotiation protocols thus creating many different combinations of algorithms and effects. The price adjustment method can be adjusted by using the FRACS variable charge_adjust_method. The resource selection criterion used by the part is set by the variable resource_selection_method. The part selection criterion used by resources is set by the variable part_selection_method. The price adjustment algorithms and different selection algorithms for each negotiation protocol are discussed in detail in [Lin, 1993].

The system also supports part-initiated negotiation protocols that do not rely on the price mechanism. It also supports dispatching rule controls. This allows the user to compare the performances of different negotiation protocols and access the effectiveness of the price mechanism.

3.3 Monitoring Interface

The monitoring interface module is used either by the job shop entities to inform the control system of the events which occur in the shop or by the control system to query the states of the entities.

Upon receiving an event, the corresponding entity in the control system either updates the information or triggers actions corresponding to the event.

When the snap_shot command is executed, all job shop entities report their states to the control module.

3.4 Control Interface

The control interface is used to control the activities of resources in the manufacturing system. The interface module includes the functions to load the part entity to machine, to move the entity from the machine to the output buffer, to load a part to an AGV, etc.

The control simulation system uses these interfaces to control the operations of the system entities. And the manufacturing environment simulation module uses these interfaces to simulate the handling and functioning of the system entities.

3.5 Communication Interface

The communication module is decomposed into two layers; the upper layer corresponds to the communication protocols that are used in the control and simulation system. The lower layer corresponds to the actual communication protocols of the underlying network (such as TCP/IP). The upper layer interface includes the functions such as send_message, receive_message, broadcast_message, etc.

The lower layer protocol depends on the actual communication package used. Possible targets include parallel communication systems such as ISIS, Presto, MPI, or loosely coupled communication systems such as TCP/IP. To port the communication module to a new type of communication system, one only needs to implement the upper layer communication modules based on the lower layer which is the interface that the target communication system provides. In our implementation of the simulation system, we simulated the lower layer communication with a simulated virtual network which transfers message among entities through an object-oriented interface.

The communication interfaces in the control and the simulation systems convert messages into events that take certain actions or redistribute the messages to entities in their domain (such as a resource agents). For example, when a part announces a ready operation, it issues a broadcasting message to all machines that are capable of performing that operation. A copy of the broadcasted message is distributed to all agents of the machines in the group which would in turn trigger an event in each agent, causing the agent to process the message.

4 EXPERIMENTS WITH THE AGENT-BASED CONTROL AND SIMULATION SYSTEM

The agent-based control and simulation system can be used for different purposes. In particular, we have used it to test

1. the performances of the proposed agent-based control framework,
2. the comparison of different negotiation protocols,
3. the comparison of negotiation control and dispatching rule control,
4. the effectiveness of different types of input process plans.
5. the effect of look ahead in negotiation, and
6. the performance comparison of different look ahead schemes.

Our results show the the proposed framework provides a coordinated information flow and physical flow and can cope with the machine failures and parts objective changes. They also show that the price mechanism reacts to the part's priority, machine capability, and system loads. We also see the resource-centered negotiation scheme achieves a slightly better overall system performance, and the part-centered negotiation scheme reacts better to the part's objective function. We also observed that the use of the flexibility embedded in the flexible processing process plan improves system performance, and a small degree of look ahead helps system performance. However, too much look ahead causes system degradation under the dynamic manufacturing environment [Lin, 1993].

The simulation system provides us with a flexible testbed such that new experiments can be easily setup. For example, when we decided to compare our negotiation protocols to that of Maley's [Maley and Solberg, 1987] which only uses part intelligence and does not use the price mechanism, we were able to implement Maley's algorithm in a few days (as opposed to months) based on the simulation system. The simulation system allows the user to concentrate on control algorithms and significantly shortens the required development time.

5 SUMMARY

In this paper, we have presented a generic agent based control simulation system. The system consists of a collection of agents who make routing and processing decisions in real time through negotiation protocol and price and objective control algorithms. Changes in the environment, objectives, or the system itself can be incorporated quickly and smoothly. Resource failures will not disrupt operation of the system. The system also allows heterogeneous job objectives, admits job priorities, recognizes multiple resource types, and allows multiple step negotiation between parts and resources to ensure system effectiveness.

The system can model generic manufacturing environment including machines with different types, capability, and input and output buffers, system layout, transporters, central buffers, different part types, multi-objective, arrival rates, and different types of input process plans. It also supports different negotiation protocols and simple dispatching control rules. Therefore, the control and simulation system can also be used to model and test different manufacturing environments, input process plans, negotiation protocols, control schemes, and price adjustment schemes. When linked with real manufacturing systems, it can also be used as a real time manufacturing control system.

ACKNOWLEDGMENTS

This work was supported in part by the Engineering Research Center for Intelligent Manufacturing Systems under a grant from the National Science Foundation, number CDR 8803017, with additional funding from the Defense Advanced Research Projects Agency (DARPA).

REFERENCES

Lin, Grace Y., and James J. Solberg. 1991. Effectiveness of Flexible Routing Control. *International Journal of Flexible Manufacturing Systems*, Vol. 3, No. 3/4, 189–212.

Lin, Grace Y., and James J. Solberg. 1992. Integrated Shop Floor Control Using Autonomous Agents. *Special Issue for Integrated Manufacturing Systems, IIE Transactions*, Vol. 24, No. 3, 57–71.

Lin, Grace Y. 1993. Distributed Production Control for Intelligent Manufacturing Systems. *Ph.D. Thesis*, Purdue University, May 1993.

Maley, James G. and James J. Solberg. 1987. Part Flow Orchestration in CIM. *Proceedings of the International Conference on Production Research*, Cincinnati, OH, 1987, 17–20.

AUTHOR BIOGRAPHIES

GRACE Y. LIN is working in the Manufacturing Research Department of IBM T.J. Watson Research Center. She Received her Ph.D. degree in Industrial Engineering from Purdue University in 1993. Her research interests include Production Planning, Scheduling, and Control, Computer Integrated Manufacturing, and Supply Chain Management. She was one of the recipients of the 1994 IIE Doctoral Dissertation Award. Her publications have appeared in the Journal of Flexible Manufacturing Systems, IIE Transactions, and some major conference proceedings. She is a member of IIE, ORSA, and TIMS.

JAMES J. SOLBERG is the Director of the Engineering Research Center for Intelligent Manufacturing Systems and Ransburg Professor of Manufacturing and Industrial Engineering at Purdue University in West Lafayette, Indiana. He received his degrees from Harvard and The University of Michigan. In addition to several teaching awards, Dr. Solberg won the "Book of the Year" Award in 1977 and the David F. Baker Distinguished Research Award from the Institute of Industrial Engineers in 1982. He is member of the National Academy of Engineering. As director of the Engineering Research Center for Intelligent Manufacturing Systems since its formation in 1985, Dr. Solberg manages a cross-disciplinary program in research, education, and technology transfer involving some 60 companies, 40 professors and 200 students.