Protocol Design and Performance Analysis for Manufacturing Message Specification: A Petri Net Approach

F.-Y. Wang, *Member, IEEE*, K. Gildea, H. Jungnitz, and D. D. Chen

Abstract— **The Manufacturing Message Specification (MMS) is the ISO standard communication protocol specific to manufacturing. To analyze MMS design and performance, service unit automata are introduced to represent individual MMS services, while service connection Petri Nets (PN's) are constructed from these automata to describe MMS service connections and processes. This approach makes MMS protocol specification and analysis possible in terms of well-developed concepts and methods in PN theory. It leads to a distributed and hierarchical model of MMS software system by integrating service connection PN's. A generalized stochastic PN for MMS performance evaluation is obtained by incorporating service parameters and time factors into the model. A technique based on T-invariants is used to simplify the performance analysis.**

I. Introduction

TO ACHIEVE fast market response, efficiency, quality, and cost effectiveness, there is an increasing emphasis on integration in today's manufacturing environments. However, automated machines and processes found on the shop floor today are usually either supplied by many different vendors or developed specifically for applications at hand. Likewise, the information systems, such as manufacturing resource planning, shop floor control, and computer-aided design and manufacturing systems, which manage and control these machines and processes, run on computers from different manufacturers. Integrating these automated machines and processes, as well as supporting information systems by providing a means to share timely, accurate data promises substantial gains in manufacturing productivity.

To facilitate integration, much effort has been used to standardize communication interfaces among computers and industrial devices. The most notable is ISO's work on Open Systems Interconnection (OSI) [1]. OSI is concerned with the exchange of information between application processes running in systems by different manufacturers. The result of this effort is a set of standard protocols organized into seven functional layers: application, presentation, session, transport, network, data link, and physical layers.

Manuscript received October 21, 1993; revised May 5, 1994.

F.-Y. Wang and D. D. Chen are with the Department of Systems and Industrial Engineering, University of Arizona, Tucson, AZ 85721 USA.

K. Gildea is with Power Parallel Systems, IBM Corporation, Kingston, NY 12401 USA.

H. Jungnitz is with Mercedes-Benz AG, MEP/TB, 70302 Stuttgart, Germany.

IEEE Log Number 9405104.

The ISO standard protocol specific to manufacturing is Manufacturing Message Specification (MMS) [2]. MMS is an application layer protocol of Manufacturing Automation Protocol [3] that allows host computers to communicate with robots, NC machines, programmable controllers, and other factory devices. To be general yet flexible to accommodate many types of devices, MMS accommodates specific companion standards. Companion standards are currently being developed for NC machines, programmable controllers, robots, and semiconductor equipments.

Design and implementation of reliable and efficient protocols require formal models to be used for analysis and evaluation. During the past decade, many works have shown that PN based models are very useful for this purposes. Burkhardt *et al.* have established a model for OSI Communication Services and Protocol using *Predicate/Transition Net* (PTN) and applied the results successfully to obtain a specification of the OSI-Transport and OSI-Network Services as well as for the specification of the OSI-Transport Protocol classes 2 and 3 [4]. Another extension of the PN model, the *Numerical Petri Net* (NPN), has been employed by Billington to specify the OSI-Transport Service (TS) [5], [6]. The results show that the NPN technique allows all interactions between the TS-provider and its users to be specified for a single connection and provides a very simple description of the queues used to model the interactions between TS-users on a connection. It was shown that NPN's provide a visually appealing model of signal flow, which facilitates the design process and enables a complete and unambiguous specification to be derived.

For PTN and NPT models, however, it is difficult, if not impossible, to use analytic methods to analyze the behavior of systems. E.g., there is no obvious way to incorporate memory reference enabling conditions and transition operations in NPN into PN analysis methods. To apply concepts and methods in PN's for design and analysis, we had used the rigorous PN models to describe the Connection Management Services (CMS) and Confirmed Services (CS) in MMS [7]. Based on the PN models, invariant analysis has led to the analytic derivation of service process properties of CMS and the detection of an undesirable property contained in CS specification [7], [8]. These results are very helpful for understanding MMS service behaviors and implementation.

The objective of this paper is to develop a formal PN-based methodology of for both protocol analysis and performance

evaluation of MMS. We will generalize our previous works developed in [9], [10] to describe MMS service units and their connections with standard PN's and to conduct MMS performance evaluation with generalized stochastic Petri nets (GSPN). This approach can support the portability, extendibility, and usability of MMS software by providing a uniform, complete, unambiguous, graphic, and mathematical representation for MMS design and implementation. Combined with other applications of PN's in modeling, control, and evaluation of manufacturing systems [11]–[14], it can lead to an integrated modeling of task execution and communication protocol in manufacturing environments.

II. Specification of Service Connections

Currently, there are 11 types of services in MMS (including the file access service and the file management services). Each MMS service is required to be able to process both service requesting and responding. MMS standard documentation [2] has specified at various levels the service state transition diagrams for service requesters and/or responders. Since a MMS-user has to play the roles of both service requester and responder, its service state transition diagrams for the two roles should be combined together. To this end, we introduce a special class of automata of symmetric structure, called service unit automata, to characterize MMS services. Based on this service specification, a formalism to build ordinary PN's to represent the service connection between two MMS-users and Colored Petri Nets (CPN's) for multiple service connection among many MMS-users will be constructed.

A. Service Unit Automata

To formalize the specification of the service state transition diagrams, we introduce the following.

Definition 1: An automaton $S = (Q, \Sigma, \delta, q_0)$ is called a *service unit automaton* (SUA) if $Q = Q_a \cup Q_b \cup Q_c$ and $\Sigma = \Sigma_a \cup \Sigma_b \cup \Sigma_c$, where $Q_a \cap Q_b = Q_b \cap Q_c = Q_c \cap Q_a = \phi$ and $\Sigma_a \cap \Sigma_b = \Sigma_b \cap \Sigma_c = \Sigma_c \cap \Sigma_a = \phi$, satisfy the following conditions:

1. *Symmetric Structure*: (i) $q_0 \in Q_c$, $Q_a = \{q_{a1}, \ldots, q_{ak}\}$, $Q_b = \{q_{b1}, \ldots, q_{bk}\}$; (ii) $\Sigma_a = \Sigma_{as} \cup \Sigma_{ar}$, $\Sigma_b = \Sigma_{bs} \cup \Sigma_{br}$, $\Sigma_{as} \cap \Sigma_{ar} = \Sigma_{bs} \cap \Sigma_{br} = \phi$, and $\Sigma_{as} = \{e_{as1}, \ldots, e_{asm}\}$, $\Sigma_{ar} = \{e_{ar1}, \ldots, e_{arn}\}$, $\Sigma_{bs} = \{e_{bs1}, \ldots, e_{bsn}\}$, $\Sigma_{br} = \{e_{br1}, \ldots, e_{brm}\}$.
2. *Transition Consistency*: (i) For any $e \in \Sigma_c$, if $\delta(q, e)$ is defined, then $\delta(q, e) \in Q_a \cup Q_c$ if $q \in Q_a \cup Q_c$ or $\delta(q, e) \in Q_b \cup Q_c$ if $q \in Q_b \cup Q_c$; (ii) For any $e \in \Sigma_a$, if $\delta(q, e)$ is defined, then $\delta(q, e) \in Q_a \cup Q_c, q \in Q_a \cup Q_c$; and (iii) For any $e \in \Sigma_b$, if $\delta(q, e)$ is defined, then $\delta(q, e) \in Q_b \cup Q_c, q \in Q_b \cup Q_c$.
3. *Initial Condition*: (i) If $\delta(q_0, e)$ is defined and $e \in \Sigma_a$, then $e \in \Sigma_{as}$; (ii) If $\delta(q_0, e)$ is defined and $e \in \Sigma_b$, then $e \in \Sigma_{br}$.

Remark 1: In an SUA, Q_a represents the set of possible service states when the SUA plays the role of a service requester, Q_b the set of possible service states when the SUA plays the role of a responder, Q_c the set of possible service states shared by both requester and responder.

Remark 2: Similarly, Σ_a represents the set of possible service transitions (or primitives) when the SUA is a service requester, Σ_b the set of possible service transitions when the SUA is a service responder, Σ_c the set of possible service transitions used by both. Σ_{as} represents the service-request transitions which send a message and Σ_{ar} the service-request transitions which receive a message. Σ_{bs} and Σ_{br} have the same interpretation with respect to the service-response transitions.

Remark 3: The condition of symmetric structure implies that for each service-request state, there corresponds a service-response state; and for each service-request transition which sends a message, there corresponds a service-response transition which receives a message and vice versa.

Remark 4: The condition of transition consistency indicates that the service-request transitions will always keep the SUA in a service state of service requester, and that the service-response transitions will always keep the SUA in a service state of service responder.

Remark 5: The initial condition requires that only the service-request transitions which send a message, and the service-response transitions which receive a message may be initially enabled. It can be shown that these conditions conform with the proposed MMS service models in [2] and that all the MMS services which require confirmation can be described by service unit automata. Therefore, *SUA model provides a uniform and unambiguous representation for MMS services.*

To insure that a sequence of service requests can always be served, and a sequence of service responses will always correspond to some service requests, we introduce the service compatible condition. To this end, the following notations are introduced: for any state $q \in Q$, its co-state is defined as:

$$\overline{q} = q \text{ if } q \in Q_c, \text{ or } q_{bi} \text{ if } q = q_{ai}, \text{ or } q_{ai} \text{ if } q = q_{bi};$$

for any transition $e \in \Sigma$, its co-transition is

$$\overline{e} = e \text{ if } e \in \Sigma_c, \text{ or } e_{bri}/e_{bsi} \text{ if } e = e_{asi}/e_{ari},$$
$$\text{or} \quad e_{ari}/e_{asi} \text{ if } e = e_{bsi}/e_{bri};$$

for any sequence of transitions $x = e_1 \cdots e_s \in \Sigma^*$, its co-sequence of transitions is $\overline{x} = \overline{e}_1 \cdots \overline{e}_s$.

Now the *service compatible condition* can be specified as:

Definition 2: A SUA is said to be *service compatible* if and only if

$$\text{For any } x \in (\Sigma_a \cup \Sigma_c)^* \cup (\Sigma_b \cup \Sigma_c)^*$$
$$\text{and} \quad p \in Q\text{: } q = \delta(p, x) \Leftrightarrow \overline{q} = \delta(\overline{p}, \overline{x}).$$

Note that most of the MMS services satisfy the service compatible condition.

B. Service Connection Petri Nets

Since a service connection is established only between the two peer services, we introduce the following definition to formalize the concept of peer services in terms of SUA's.

Definition 3: Two SUA's $S_1 = (Q_1, \Sigma_1, \delta_1, q_0^1)$ and $S_2 = (Q_2, \Sigma_2, \delta_2, q_0^2)$ are said to be in the *same class of service units* if they are isomorphic, i.e., there exists a 1-1 mapping $\varphi = (\varphi_Q, \varphi_\Sigma)$ such that φ_Q: $Q_1 \rightarrow Q_2$ and φ_Σ: $\Sigma_1 \rightarrow \Sigma_2$ satisfy,

(i) $\varphi_Q(q_0^1) = q_0^2$, $\varphi_Q(q_{ci}^1) = q_{ci}^2$, $\varphi_Q(q_{ai}^1) = q_{ai}^2$, $\varphi_Q(q_{bi}^1) = q_{bi}^2$;
(ii) $\varphi_\Sigma(e_{ci}^1) = e_{ci}^2$, $\varphi_\Sigma(e_{asi}^1) = e_{asi}^2$, $\varphi_\Sigma(e_{ari}^1) = e_{ari}^2$, $\varphi_\Sigma(e_{bsi}^1) = e_{bsi}^2$, $\varphi_\Sigma(e_{bri}^1) = e_{bri}^2$;
(iii) $\varphi_Q(\delta_1(q^1, e^1)) = \delta_2(\varphi_Q(q^1), \varphi_\Sigma(e^1))$.

In other words, all the service units in the same class have the identical structure and the same state transitions, therefore, they represent peer services.

The service connection between two service units now can be formally specified by the following.

Definition 4: If S_1 and S_2 are two SUA's in the same class, the *service connection Petri net* (SPCN), $PN(S_1, S_2)$, associated with S_1 and S_2 is defined to be

$$PN(S_1, S_2) = (P, T, I, O, m_0) \quad \text{where}$$

(i) Places and Transitions:

$$P = Q_1 \cup Q_2 \cup C_1 \cup C_2,$$
$$C_i = \{p_{s1}^i, \ldots, p_{sm}^i, p_{r1}^i, \ldots, p_{rn}^i\},$$
$$T = \Sigma_1 \cup \Sigma_2;$$

(ii) Input and Output Functions:
$I(p,t) = 1$ if $\delta_k(p,t)$ is defined; or $(p,t) = (p_{si}^1, e_{bri}^2)$ or (p_{si}^2, e_{bri}^1) or (p_{ri}^k, e_{ari}^k);
$O(p,t) = 1$ if $p = \delta_k(q,t), q \in Q_1 \cup Q_2$, or $(p,t) = (p_{ri}^1, e_{bsi}^2)$ or (p_{ri}^2, e_{bsi}^1) or (p_{si}^k, e_{asi}^k); otherwise, $I(p,t) = O(p,t) = 0$, where $k = 1$ or 2.

(iii) Initial Mapping: $m_0(p) = n_i$ if $p = q_0^i$, 0 otherwise.

Remark 1: Since a PN cannot represent the "OR" condition, it is assumed, without the lost of generality in this definition, that if both $\delta(q_1, e_1)$ and $\delta(q_2, e_2)$ are defined in a SUA, then $q_1 \neq q_2 \rightarrow e_1 \neq e_2$.

Remark 2: PN(S_1, S_2) is constructed by connecting S_1 and S_2 through additional places C_1 and C_2. These new places represent the communication points (or ports) between the two units. Actually this is the only physical feasible connection mechanism to link S_1 and S_2 since one service unit cannot get access to the internal states of another unit. Therefore message exchanges between the two units can be achieved only through the communication points.

Remark 3: Using the execution rule of PN's, the SPCN PN(S_1, S_2) will specify all the possible sequences of service primitives (or transitions) occurring during a service connection between S_1 and S_2, where n_i represents the maximum number of outstanding services allowed by S_i. Therefore, SPCN PN(S_1, S_2) *defines the communication protocol between two service units.*

C. Colored Petri Nets for Multiple Service Connection

The SPCN described above can specify the service connection between two MMS-users through the execution rule of PN's. To model service connections among more than two MMS-users, however, CPN's will be much more efficient than PN's. This is due to the fact that a large number of new places and transitions are required if PN's is used to model connections. CPN's were developed for the purpose of shortening the description and analysis of the systems consisting of a large number of different processes having a similar structure and behavior [15]. Therefore it is natural to use CPN's to specify multiple service connections.

Definition 6: Let N SUA's $S_1, \ldots,$ and S_N be in the same class with a SUA $S = (Q, \Sigma, \delta, q_0)$. The multiple service connection Petri net MSCPN, CPN$(S_1, \ldots, S_N)$, associated with $S_1, \ldots,$ and S_N is a CPN defined to be

$$CPN(S_1, \ldots, S_N) = (P, T, M, I, O, m_0) \quad \text{where}$$

(i) Places, Transitions, and Colors:

$$P = Q \cup C_M, C_M = \{p_{s1}, \ldots, p_{sm}, p_{r1}, \ldots, p_{rn}\}, T = \Sigma,$$
$$\text{and} \quad M = \{c_1, \ldots, c_N\};$$

(ii) Color Functions:

$$C(q_0) = M, C(p) = CNN \text{ for all } p \in P, p \neq q_0,$$
$$\text{and} \quad C(t) = CNN \text{ for all } t \in T,$$
$$CNN = \{(c_i, c_j) \mid i \neq j, c_i \text{ and } c_j \in M\};$$

(iii) Input and Output Functions: the input and output functions I and O are defined in the same way as for SPCN and the colors associated with them are given in Table I for both the input and output colors, in which

$$REQ\colon CNN \rightarrow M, REQ((c_i, c_j)) = c_i;$$
$$RSP\colon CNN \rightarrow M, RSP((c_i, c_j)) = c_j;$$
$$ID\colon CNN \rightarrow CNN, ID((c_i, c_j)) = (c_i, c_j);$$
$$REV\colon CNN \rightarrow CNN, REV((c_i, c_j)) = (c_j, c_i);$$

(iv) Initial Mapping:

$$m_0(q_0)(c_i) = n_i, \quad i = 1, \ldots, N, \quad m_0(p) = 0, \text{ otherwise.}$$

Remark 1: In MSCPN CPN$(S_1, \ldots, S_N)$, color set M represents the set of SUA's to be connected. CNN is the set of *service connection pairs*, i.e., (c_i, c_j) means S_i is calling S_j, and n_i specifies the maximum number of outstanding services allowed by S_i. In MMS, the colors of initial places correspond to the invoking ID's for service instances. The function REQ returns the service requester of a connection pair, RSP returns the service responder, and REV changes the service role. REV is used when a service responder enters or leaves a service state with no distinction between a service requester and a service responder.

Remark 2: Similarly as for SPCN, using the execution rule of CPN's, the MSCPN CPN$(S_1, \ldots, S_N)$ will specify all the possible sequences of service primitives occurring during service connections among $S_1, S_2, \ldots,$ and S_n. Therefore, CPN *defines the communication protocol between service units* $S_1, \ldots, S_N$.

III. Analysis of Service Processes

Connection PN's MSCPN's or SPCN's provide a complete, unambiguous, and mathematical specification for the communication protocol between service units. In this section we will apply concepts and methods developed in PN theory to analyze service connection processes.

Considering S_1 and S_2 as the subnets of $\mathrm{PN}(S_1, S_2)$, we see from the construction of $\mathrm{PN}(S_1, S_2)$ that both S_1 and S_2 are the closed subnets of $\mathrm{PN}(S_1, S_2)$. Since the original service units are always bounded, the service connection between S_1 and S_2 has the following property:

Lemma 1: Consider S_1 and S_2 as two subnets of $\mathrm{PN}(S_1, S_2)$, then both S_1 and S_2 are conservative, hence, are bounded subnets of $\mathrm{PN}(S_1, S_2)$.

This lemma indicates that the number of services in S_1 or S_2 is bounded during the service connection specified by $\mathrm{PN}(S_1, S_2)$. However, in general, SPCN $\mathrm{PN}(S_1, S_2)$ is neither conservative nor bounded. The number of tokens in places C_1 or C_2 might be unbounded for certain connections.

Another property of $\mathrm{PN}(S_1, S_2)$ is the service compatibility. That is, if both SUA's are service compatible, each service-request transition sequence from one SUA can be responded by a corresponding service-response transition sequence from the other SUA, and that every service-response transition sequence from one SUA will always be caused by a corresponding service-request transition sequence from the other. Formally, we have the following.

Lemma 2: Let $\varphi = (\varphi_Q, \varphi\Sigma)$ be the isomorphic mapping between two service compatible SUA, S_1 and S_2. Consider S_1 and S_2 as the subnets of $\mathrm{PN}(S_1, S_2)$ and define the state transitions of S_1 and S_2 with respect to the execution of $\mathrm{PN}(S_1, S_2)$, then,

(i) For any $x \in \Sigma_1^*$ and $p \in Q_1$: $\overline{q} = \delta_1(\overline{p}, \overline{x}) \Leftrightarrow \varphi_Q(q) = \delta_2(\varphi_Q(p), \varphi_\Sigma(x))$;

(ii) For any $x \in \Sigma_2^*$ and $p \in Q_2$: $\overline{q} = \delta_2(\overline{p}, \overline{x}) \Leftrightarrow \varphi_Q(q) = \delta_1(\varphi_Q(p), \varphi_\Sigma(x))$.

The proof of this lemma can be achieved by performing induction on the length of x.

Finally, we investigate the service process property from the aspect of liveness/deadlock-free of the SCPN. A deadlock in a PN occurs when a state is reached where no transitions in the net can be fired from that point on, and a PN is live if from any state reached from the initial state, it is possible to fire any transition in the net [16]. Liveness guarantees the absence of deadlocks. In general, it can be shown that the SCPN of two SUA's may lead to connection deadlock, therefore additional constraints have to be imposed on the construction of SUA's to insure the deadlock-free and/or liveness in the service processes. Based on the analysis given by [7], a simple way to impose the required constraints is to use the common abort service transition, that is,

Definition 5: An SUA is said to have the complete abort transition if for any $q \in Q$ and $q \neq q_0$, there is $e \in \Sigma_c$ such that $\delta(q, e) = q_0$.

Most of the MMS services have the complete abort transition. Obviously, a SUA with the complete abort transition is deadlock free if there are some initially enabled transitions. The following theorem further indicates that if the two SUA's are live, then their CSPN is live too.

Theorem 3: If both S_1 and S_2 are service compatible and have the complete abort transition, then, the SCPN $\mathrm{PN}(S_1, S_2)$ contains no deadlock. Further, if both S_1 and S_2 are live, then $\mathrm{PN}(S_1, S_2)$ is a live PN.

Proof: Let $R(\mathrm{PN}, m_0)$ be the reachability set of $\mathrm{PN}(S_1, S_2)$, $R(S_i, m_{0i})$ be the reachability set of S_i when S_i is considered as an independent PN, $i = 1, 2$. For each $m \in R(\mathrm{PN}, m_0)$, let m_i be the restriction of m on $S_i, i = 1, 2$. Clearly, $m_i \in R(S_i, m_{0i})$.

First of all, $\mathrm{PN}(S_1, S_2)$ contains no deadlock. This is due to the fact that abort transitions are in $\Sigma_c^1 \cup \Sigma_c^2$, thus according to Definitions 1 and 4, they do not have the input places from C_1 or C_2. Therefore, for any $m \in R(\mathrm{PN}, m_0)$, there always exists an enabled abort transition under m_i in S_i by the definition of the complete abort transition, $i = 1, 2$.

To show the liveness, for any $m \in R(\mathrm{PN}, m_0)$, let $x_i \in \Sigma_i^*$ be the sequence of transitions in S_i which transforms S_i from m_{0i} to $m_i, i = 1, 2$. For any transition $e_i \in \Sigma_i$, since S_i is live, there exists a sequence $y_i \in \Sigma_i^*$ such that e_i is enabled after firing y_i from m_i when S_i is considered as an independent PN. Since both S_1 and S_2 are service compatible and have the complete abort transition, according to Lemma 2, we can always arrange x_i and m_i such that,

$$\overline{x}_1 = \varphi_\Sigma(x_2) \text{ and } \overline{m}_1 = \varphi_Q(m_2),$$

where $(\varphi_Q, \varphi_\Sigma)$ is the isomorphic mapping between S_1 and S_2. Therefore, e_i will be enabled by firing both y_i in one SUA and z_i in the other SUA, where z_i is given by $\overline{z}_i = \varphi_\Sigma(y_i)$. Hence, $\mathrm{PN}(S_1, S_2)$ is a live PN. This completes the proof.

This theorem ensures that all service primitives can be fired from any service state during the service connection defined by $\mathrm{PN}(S_1, S_2)$.

It can be shown easily that analysis for multiple service connection can be performed using $\mathrm{CPN}(S_1, \ldots, S_N)$ in the same way as using $\mathrm{PN}(S_1, S_2)$ for the service connection between two SUA's. All the conclusions obtained above are still valid in the case of multiple service connections.

IV. MMS Software Specification and Architecture

MMS software architecture consists of three major components: an interface from the application process to a MMS provider, the MMS provider, and an interface to a Common Application Service Element (CASE) provider. The MMS provider is defined as that part of the application entity that conceptually provides the MMS services through the exchange of MMS Protocol Data Units (PDU's). These PDU's are encoded using the Abstract Syntax Notation One (ASN.1). Hence, part of the MMS provider function is the ASN.1 encoder and decoder (i.e., parser) [9]. Fig. 1 presents a MMS architecture which only provides the environment and general management services (EGMS), the confirmed services (CS), the file management services (FMS), and the Variable Access services (VAS). This section presents the results of specifying the MMS software using the method developed in the previous sections. We will consider only EGMS, CS, and FMS.

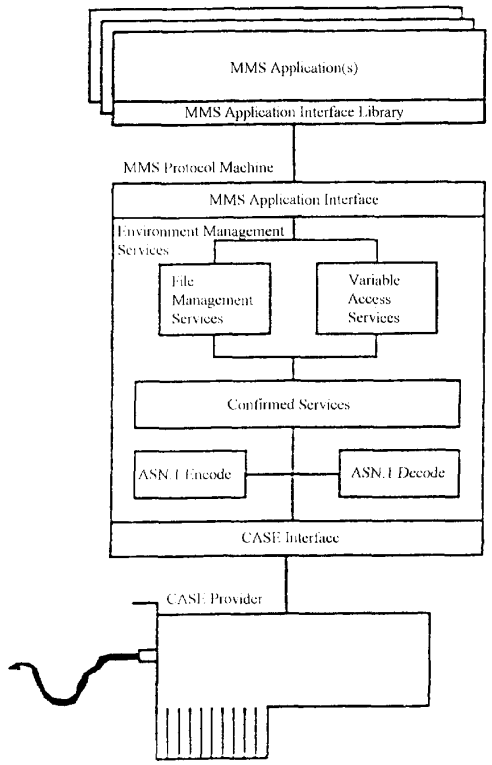


Fig. 1. MMS software architecture.

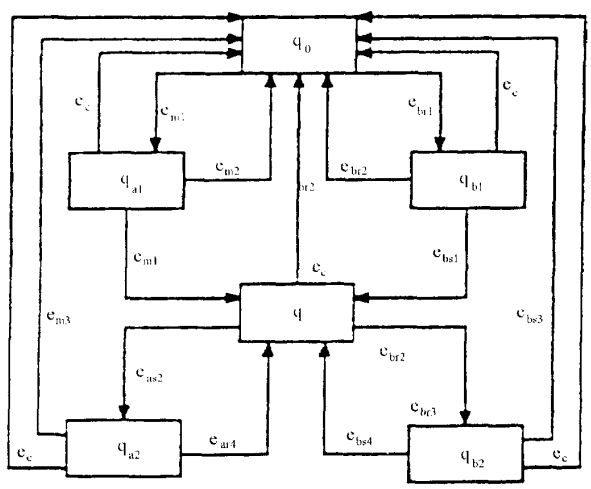


Fig. 2. Service unit automaton for EGMS.

A. Environment and General Management Services

The EGMS is used to maintain connections between MMS-users and contains the *initiate, conclude, abort, cancel, and reject* services. These services allow a MMS-user to: a) initiate communication with another MMS-user in the MMS environment, and to establish the requirements and capabilities which support that communication; b) conclude communication with another MMS-user in a graceful manner; c) abort communications with another MMS-user in an abrupt manner; d) cancel pending service requests; and e) receive notification of protocol errors that occur. The SUA for the EGMS can be specified as:

$$\begin{aligned}
&\mathrm{SUA_{EGMS}}= (Q, \Sigma, \delta, q_0), \text{ where}\\
&\quad Q = Q_a \cup Q_b \cup Q_c, \Sigma = \Sigma_a \cup \Sigma_b \cup \Sigma_c,\\
&\quad \Sigma_a = \Sigma_{as} \cup \Sigma_{ar}, \Sigma_b = \Sigma_{bs} \cup \Sigma_{br};\\
&\quad Q_a = \{q_{a1}, q_{a2}\}, Q_b = \{q_{b1}, q_{b2}\}, Q_c = \{q_0, q_c\};\\
&\quad \Sigma_c = \{e_c\text{: } \textit{abort}\},\\
&\quad \Sigma_{as}=\{e_{as1}\text{: } \textit{initiate.request}, e_{as2}\text{: } \textit{conclude.request}\},\\
&\quad \Sigma_{br} = \{e_{br1}\text{: } \textit{initiate.indication},\\
&\qquad e_{br2}\text{: } \textit{conclude.indication}\},\\
&\quad \Sigma_{bs} = \{e_{bs1}\text{: } \textit{initiate.response}+,\\
&\qquad e_{bs2}\text{: } \textit{initiate.response}-,\\
&\quad e_{bs3}\text{: } \textit{conclude.response}+, e_{bs4}\text{: } \textit{conclude.response}-\},\\
&\quad \Sigma_{ar}=\{e_{ar1}\text{: } \textit{initiate.confirm}+, e_{ar2}\text{: } \textit{initiate.confirm}-,\\
&\quad e_{ar3}\text{ : } \textit{conclude.confirm}+, e_{ar4}\text{: } \textit{conclude.confirm}-\}.
\end{aligned}$$

The service states define the restrictions on the use of MMS services:

- q_0 No MMS Environment State—while in q_0, an MMS-user may only issue the initiate.request service primitive;
- q_{a1} Establishing MMS Environment (Calling) State—while in q_{a1}, a MMS-user may only issue the abort service primitive;
- q_{b1} Establishing MMS Environment (Called) State—while in q_{b1}, a MMS-user may only issue the initiate.response or abort service primitives;
- q_c MMS Environment State: this state is divided into a series of substates described in clauses 8–16 of MMS [2]. The only events which cause an exit from q_c are the issuance of an abort.request, the issuance of a conclude.request service primitive, the receipt of an abort.indication, or the receipt of a conclude.indication service primitive;
- q_{a2} Relinquishing MMS Environment (Requester) State—while in q_{a2}, the only request primitive that the MMS-user may only issue the abort service primitive; and
- q_{b2} Relinquishing MMS Environment (Responder) State—while in q_{b2}, the MMS-user may only issue the abort and the conclude.response service primitives.

The state diagram of $\mathrm{SUA_{EGMS}}$ for the EGMS is given in Fig. 2. The SCPN $\mathrm{PN_{EGMS}}$ for EGMS between two MMS-users is given in Fig. 3. Since $\mathrm{SUA_{EGMS}}$ is service compatible and has complete abort transition, both $\mathrm{PN_{EGMS}}$ and $\mathrm{CPN_{EGMS}}$ are live PN's, therefore, deadlock-free.

As demonstrated in [7], the connection behavior of EGMS can be further explored by performing PN analysis (e.g., P-invariant and/or T-invariant analysis) on $\mathrm{PN_{EGMS}}$ or $\mathrm{CPN_{EGMS}}$.

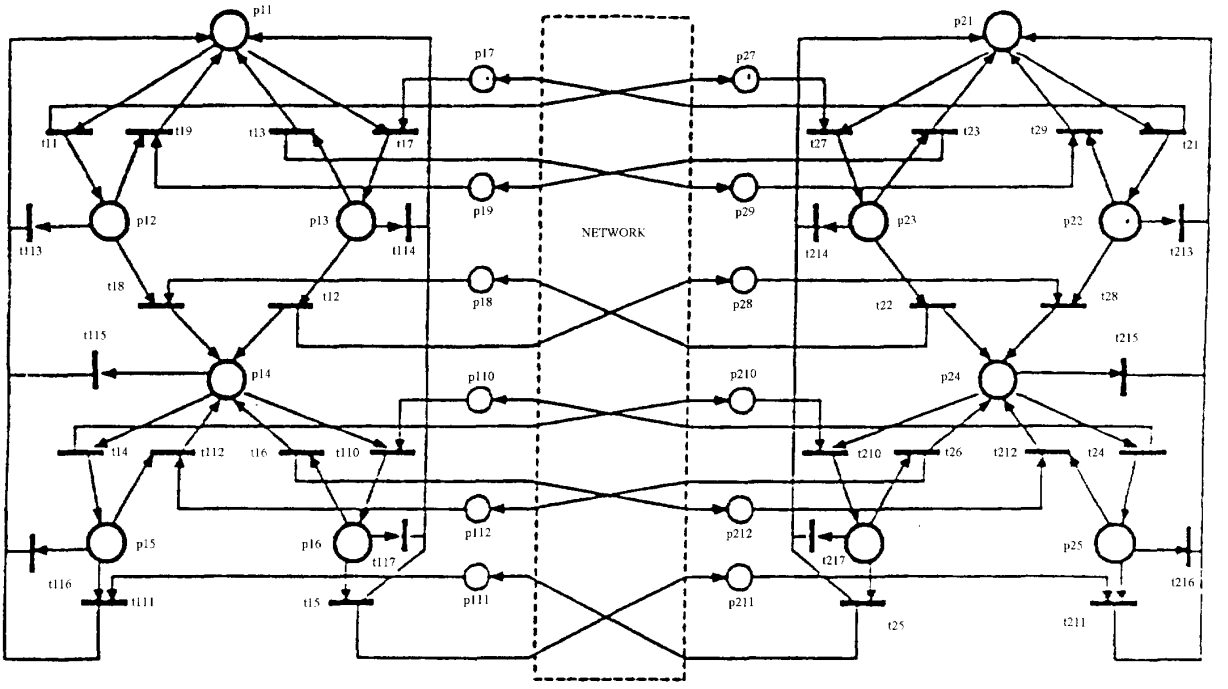


Fig. 3. Service connection Petri net for EGMS.

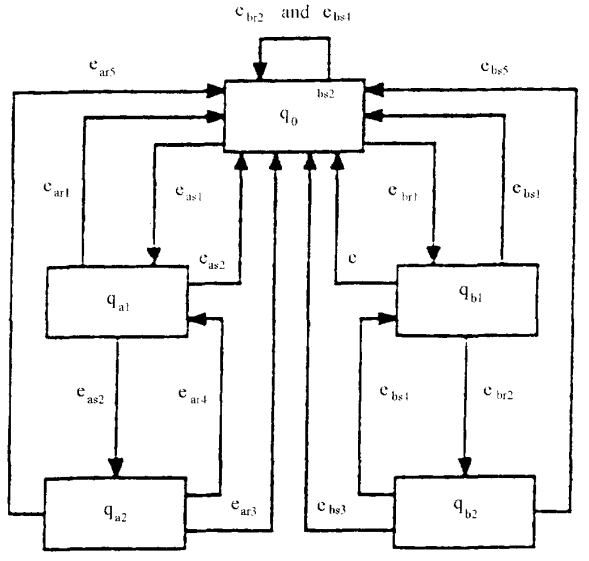


Fig. 4. Service unit automaton for CS.

B. Confirmed Services

The CS is the largest class of services in MMS. All MMS services, whether confirmed or not, are initiated when a service requester sends a service request (indication) to a service provider. What distinguishes confirmed services from unconfirmed services is the use of a service response (confirmation) by the service provider to signal the completion of the service to the requester.

The execution of any single MMS confirmed service instance is associated with the seven PDU's: 1) the Confirmed-RequestPDU, 2) the Confirmed-ResponsePDU, 3) the Confirmed-ErrorPDU, 4) the Cancel-RequestPDU, 5) the Cancel-ResponsePDU, 6) the Cancel-ErrorPDU, and 7) the RejectPDU. All these PDU's shall be sent in the same presentation context. The RejectPDU will not be considered here. The SUA for the CS can be specified as:

$$\mathrm{SUA_{CS}} = (Q, \Sigma, \delta, q_0).$$

where $Q = Q_a \cup Q_b \cup Q_c, \Sigma = \Sigma_a \cup \Sigma_b,$

$\Sigma_a = \Sigma_{as} \cup \Sigma_{ar}, \Sigma_b = \Sigma_{bs} \cup \Sigma_{br};$

$Q_a = \{q_{a1}, q_{a2}\}, Q_b = \{q_{b1}, q_{b2}\}, Q_c = \{q_0\};$

$\Sigma_{as} = \{e_{as1}$: *x.request*, e_{as2}: *cancel.request*$\}$.

$\Sigma_{br} = \{e_{br1}$: *x.indication*, e_{br2}: *cancel.indication*$\}$,

$\Sigma_{bs} = \{e_{bs1}$: *x.response+*, e_{bs2}: *x.response−*,

e_{bs3}: *cancel.response+*, e_{bs4}: *cancel.response−*,

e_{bs5}: *x.response − and_cancel.response+*$\}$,

$\Sigma_{ar} = \{e_{ar1}$: *x.confirm+*, e_{ar2}: *x.confirm−*,

e_{ar3}: *cancel.confirm+*, e_{ar4}: *cancel.confirm−*,

e_{ar5}: *x.confirm − and_cancel.confirm+*$\}$.

The service states describe the progression of a CS instance:

q_0 Service Idle State—no service primitive is issued;

q_{a1} Service Pending Requester State—receipt of a request primitive for any of the confirmed MMS services;

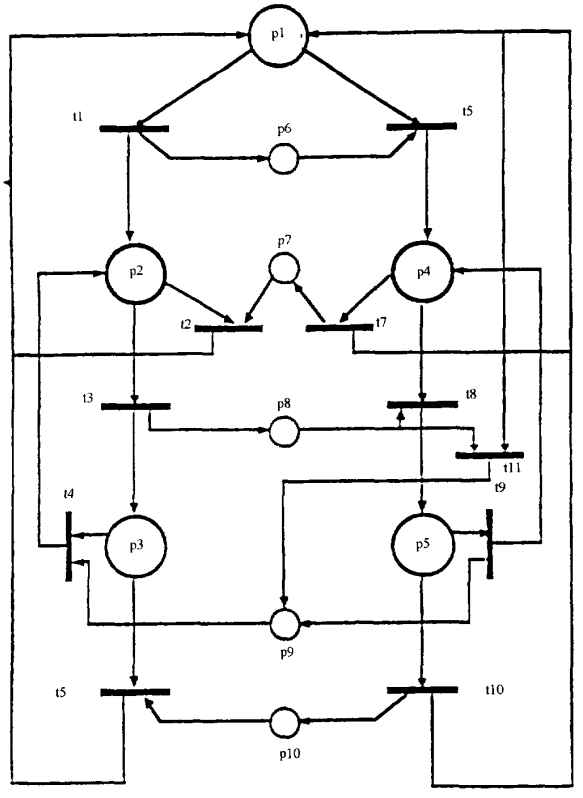


Fig. 5. Multiple service connection colored Petri net for CS.

q_{b1} Service Pending Responder State—receipt of a Confirmed-RequestPDU for any of the confirmed MMS services;

q_{a2} Canceling from Requester State—receipt of a cancel request service primitive; and

q_{b2} Canceling from Responder State—receipt of a Cancel-RequestPDU.

The state diagram of SUA_{CS} for the CS is given in Fig. 4, which is the first alternative introduced by [8] for the modification of the original CS state diagram defined in [2]. The SCPN PN_{CS} for CS between two MMS-users and the corresponding colored MSCPN CPN_{CS} can be constructed easily. Fig. 5 gives CPN_{CS}.

Note that since SUA CS is not service compatible and does not have complete abort transition, both PN_{CS} and CPN_{CS} may lead to deadlock. Dead lock occurs when the two connected MMS-users insist on sending their own service requests without responding to another's request. However, this deadlock actually will not appear in the implementation since: (i) There is a large number of initial tokens representing the outstanding services allowed at the service idle places; (ii) the initial tokens have been divided into two groups: one is only for service requests and the other only for services responses. This has been specified during the EGMS phase by the two parameters *negotiatedMaxServOutstandingCalling* and *negotiatedMaxServOutstandingCalled* of the Initiate-ResponsePDU ([2], part two). Therefore, the deadlock cases can be prevented.

C. File Management Services

FMS is the only specific-purpose MMS service considered in this work. It provides the necessary functionality for reading files containing control programs and data from filestores in control devices and file servers, and for managing filestores by enumerating the names and attributes of files, renaming, and deleting files. There are two service sequences in FMS, i.e., *FileRename* → *FileDelete* → *FileDirectory* and *FileOpen* → *FileRead* → *FileClose*. We only give the result for *FileOpen.FileRead.FileClose* service sequence here. The SUA for the FMS can be specified as:

$$
\begin{aligned}
&SUA_{FMS} = (Q, \Sigma, \delta, q_0),\\
&\text{where}\quad Q = Q_a \cup Q_b \cup Q_c, \Sigma = \Sigma_a \cup \Sigma_b \cup \Sigma_c,\\
&\qquad \Sigma_a = \Sigma_{as} \cup \Sigma_{ar}, \Sigma_b = \Sigma_{bs} \cup \Sigma_{br};\\
&\qquad Q_a = \{q_{a1}, q_{a2}, q_{a3}, q_{a4}\}, Q_b = \{q_{b1}, q_{b2}, q_{b3}, q_{b4}\},\\
&\qquad Q_c = \{q_0\}; \Sigma_c = \{e_c\text{: } \textit{abort}\},\\
&\qquad \Sigma_{as} = \{e_{as1}\text{: } \textit{FileOpen.request},\\
&\qquad\qquad e_{as2}\text{: } \textit{FileRead.request},\\
&\qquad\qquad e_{as3}\text{: } \textit{FileClose.request}\},\\
&\qquad \Sigma_{br} = \{e_{br1}\text{: } \textit{FileOpen.indication},\\
&\qquad\qquad e_{br2}\text{: } \textit{FileRead.indication},\\
&\qquad\qquad e_{br3}\text{: } \textit{FileClose.indication}\},\\
&\qquad \Sigma_{bs} = \{e_{bs1}\text{: } \textit{FileOpen.response}+,\\
&\qquad\qquad e_{bs2}\text{: } \textit{FileOpen.response}-,\\
&\qquad\qquad e_{bs3}\text{: } \textit{FileRead.response}+,\\
&\qquad\qquad e_{bs4}\text{: } \textit{FileRead.response}-,\\
&\qquad\qquad e_{bs5}\text{: } \textit{FileClose.response}+,\\
&\qquad\qquad e_{bs6}\text{: } \textit{FileClose.response}-\},\\
&\qquad \Sigma_{ar} = \{e_{ar1}\text{: } \textit{FileOpen.confirm}+,\\
&\qquad\qquad e_{ar2}\text{: } \textit{FileOpen.confirm}-,\\
&\qquad\qquad e_{ar3}\text{: } \textit{FileRead.confirm}+,\\
&\qquad\qquad e_{ar4}\text{: } \textit{FileRead.confirm}-,\\
&\qquad\qquad e_{ar5}\text{: } \textit{FileClose.confirm}+,\\
&\qquad\qquad e_{ar6}\text{: } \textit{FileClose.confirm}-\}.
\end{aligned}
$$

The service states describe the progression of a FMS instance:

q_0 File Close State—no file service primitive is issued;

q_{a1} File Open Pending (Requester) State—wait for FileOpen confirmation;

q_{b1} File Open Pending (Responder) State—receipt of a FileOpen indication;

q_{a2} File Open (Requester) State—FileRead ready from file requester;

q_{b2} File Open (Responder) State—FileRead ready from file responder;

q_{a3} File Reading (Requester) State—wait for FileRead confirmation;

q_{b3} File Reading (Responder) State—receipt of a FileRead indication;

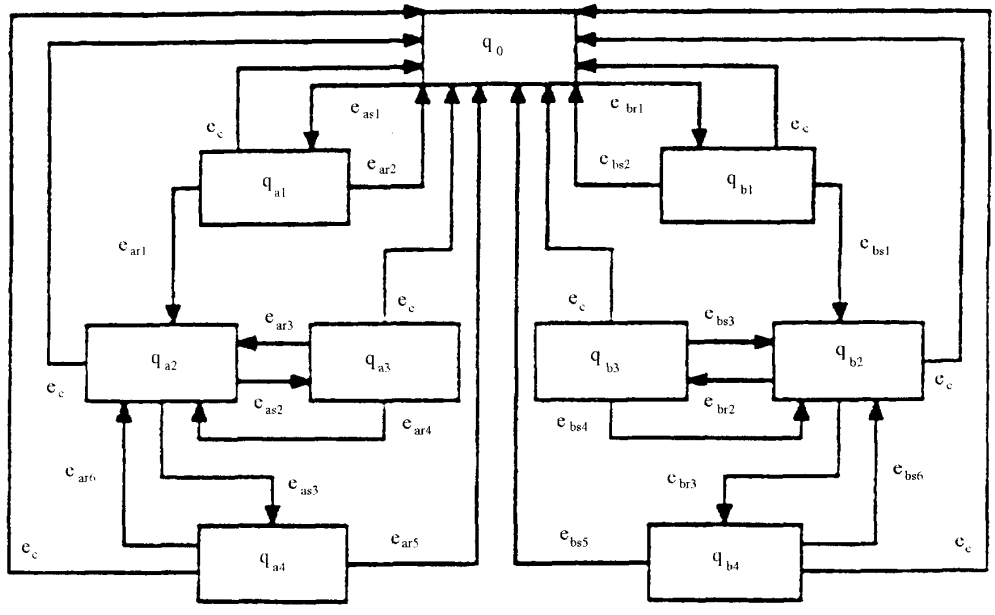


Fig. 6. Service unit automaton for FMS.

q_{a4} File Close Pending (Requester) State—wait for File-Close confirmation; and

q_{b4} File Close Pending (Responder) State—receipt of a FileClose indication.

The state diagram of SUA_{FMS} for the FMS is given in Fig. 6, from which the SCPN PN_{FMS} and MSCPN CPN_{FMS} for FMS can be constructed similarly as in the previous examples.

D. Hierarchical Petri Net Description for MMS Architecture

A hierarchical PN model for the entire MMS software can be established once we have individual PN's. In this hierarchical description the place "MMS Environment" in EGMS is replaced by a number of MMS services. Each of the services, e.g., an FMS, is defined by a PN which can communicate with its peer service PN through the CS connection PN PN_{CS} or CPN_{FMS}. Once the MMS Environment place in EGMS is invoked (i.e., tokens have arrived), one of the service PN's will be executed accordingly. Based on the PN models, a uniform service control strategy or procedure can be developed for all the services in this MMS hierarchy.

V. Performance Evaluation and Sensitivity Analysis

Theoretically, by assigning execution time distributions to transitions, the hierarchical PN for MMS can be converted into a Stochastic PN and then the performance analysis for MMS software can be conducted then on this timed net [17]. However, in this case, due to the limitation of the computation capacity, the analysis can be done only by the brute-force simulation. Thus, a simplified model has to be constructed if an analytic analysis is to be attempted. Since in MMS software, all services have to be handled by the CS unit, the PN for the CS is thus the critical section in MMS performance evaluation. Therefore, a reasonable model of performance analysis for MMS can be constructed from the PN for the CS.

A. GSPN Model Based on Confirmed Services

Fig. 7 presents the GSPN model for performance analysis developed from the PN for the CS between two MMS users. For clarity, the net is shown without place channel. Place channel N_i has an incoming arc from each of transitions n_{ij} and outgoing arcs to each of transitions n_i. The number of tokens C in this place indicates the number of communication channels available between the two users. The arrival of service activities to be processed by the CS has been modeled by a simple queue (represented by t_Q, Q_1 and Q_2). The initial number of tokens K in place Q_2 represents the maximum number of services allowed to enter the CS. The number of outstanding services allowed in MMS is indicated by the initial number of tokens S in places p_{s1} and p_{s2}. The net consists of three parts: the service requester (place p_{s1} to p_4), the responder (places p_{s2} to p_8), and the communication network (all other places). A detailed description of the net has been in the Appendix. The package SPNP has be used for the performance analysis.

B. Invariants of the GSPN

There are two reasons for performing the structural analysis for the GSPN model: examination of the boundedness and liveness of the net, and determination of the proper sequences of transition firings for the correct communication. To this end, the P-invariants and the T-invariants of the net are determined by using the package GreatSPN 1.5.

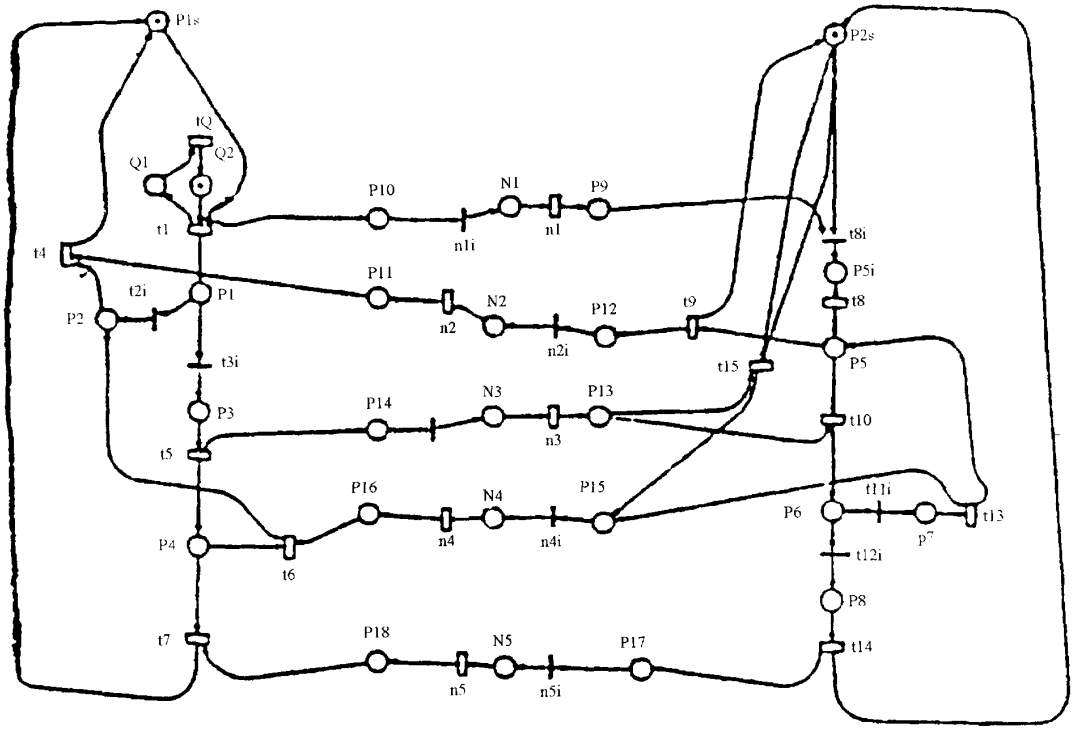


Fig. 7. The GSPN model for the MMS performance analysis.

Six minimal P-invariants have been obtained for the GSPN. The GSPN is bounded since all the places are covered by some P-invariants. The following are the first four P-invariants:

$$P_1\colon m(Q_1) + m(Q_2) = K, \quad P_2\colon m(p_{s1}) + \sum_{k=1}^{4} m(p_k) = S,$$

$$P_3\colon m(p_{s2}) + \sum_{k=5}^{8} m(p_k) + m(p_{5i}) = S,$$

$$P_4\colon m(channel) + \sum_{k=1}^{5} m(N_k) + m(p_{5i}) = C$$

The meaning of P_1 is obvious. P_2 and P_3 indicate the subnets of the GSPN corresponding to the service requester and responder are strictly conservative. P_4 represents the mutual exclusion of the communication channel resource. Also, the GSPN is live. Thus, there is no deadlock marking.

T-invariants can be used to determine the normal service operations. A *normal service cycle* is defined to be any sequence of transition firings which leads the GSPN from the initial marking and returns it back to the initial marking. There are four minimal T-invariants in the GSPN, and correspondingly, four types of minimal normal service cycles. The following firing sequences are four typical minimal normal service cycles between the service requester and responder (immediate transitions are omitted):

$$T_1\colon t_Q t_1 n_1 t_8 t_9 n_2 t_4,$$
$$T_2\colon t_Q t_1 n_1 t_8 t_5 n_3 t_{10} t_{13} n_4 t_6 t_9 n_2 t_4,$$
$$T_3\colon t_Q t_1 n_1 t_8 t_5 t_9 n_2 n_3 t_{15} n_4 t_6 t_4,$$
$$T_4\colon t_Q t_1 n_1 t_8 t_5 n_3 t_{10} t_{14} n_5 t_7.$$

T_1 specifies the type of standard service executions in which no service canceling requests have been issued by the requester. T_2 gives the type of unsuccessful canceling service executions in which canceling requests have been issued by the requester but denied later by the responder for some reason. The type of service executions represented by T_3 is a cancel-error execution in which service canceling requests from the requester are received by the responder after the services requested have already been completed. Finally, the service executions defined by T_4 are these successful canceling service executions in which canceling requests have been issued and accepted.

C. Performance Indices and T-Invariant Technique

The performance indices specify the criteria for evaluating the performance of MMS software, and therefore give useful quantitative information about the operation of MMS. However, the most important significance of those indices is that they provide guidelines for the optimum implementation of MMS software. This is achieved by examining the sensitivity of the performance indices over the range of certain system

parameters and thus finding the critical parameters on MMS performance. Specifically, the effects of the following factors on the performance of MMS have been investigated here:

- *Parser Design*: the sensitivity of MMS performance over the encoding and decoding times of PDU's. This will give us the information about how much effort should be spent on the parser design. E.g., whether it is worthwhile to try a more complex and faster parsing algorithm.
- *Communication Network*: the sensitivity of the performance over the usage of different communication networks for the MMS system. This will provide us the information about what kind of communication network that should be selected. A fast and dedicated network is alway desirable, but it might not be cost effective.
- *Application Environment*: the sensitivity of performance over the different environments in which MMS is going to be applied. This will indicate whether the MMS can have satisfactory performance, or for a given application environment, what kind of requirements should be imposed on MMS system.

For these purposes, the following performance indices have been defined for MMS analysis.

1. Average Number of Waiting
   Services (WS): $\mathrm{WS} = E[m(Q_1)]$ (1)
2. Average Number of Processing
   Services (PS): $\mathrm{PS} = \sum_{k=1}^{8} E[m(p_k)]$ (2)
3. Average Number of Transporting
   Services (TS): $\mathrm{TS} = \sum_{k=1}^{5} E[m(N_k)]$ (3)
4. Average Service Waiting
   Time (WT): $\mathrm{WT} = \mathrm{WS}/\lambda_W^e$ (4)

where $\lambda_W^e = \lambda_Q \times$ Prob[t_Q is enabled].

For the average service processing and transporting times, to avoid the complex analysis and save computation time, a technique for performance evaluation based on T-invariants will be used here. The basic idea of the T-invariant technique is: since the service executions of MMS have to follow the transition sequences specified by the T-invariants, the performance of MMS in the general case of operations can be determined by considering only the performance under the special case of operations defined by the T-invariants. The main reason that this technique is preferred is because it is very easy to calculate the performance of MMS being operated according to individual T-invariants. Clearly, when the performances of MMS under individual T-invariants have been determined, the performance in the general case be found by adding up the individual performance indices weighted by the probabilities of executing the corresponding T-invariant operations. Specifically, we have the following results:

5. Average Service Processing
   Time (PT): $\mathrm{PT} = \sum_{k=1}^{4} \alpha_k D_{sk}$ (5)
6. Average Service Transporting
   Time (TT): $\mathrm{TT} = \sum_{k=1}^{4} \alpha_k D_{tk}$ (6)

where α_k is the ratio of applying a T_k service operation: $\alpha_k = r_k / \sum_{i=1}^{4} r_i, k = 1, \ldots, 4, (r_1, r_2, r_3, r_4) =$ throughput of $(t_4, t_{13}, t_{15}, t_7)$. This computation procedure is due to the fact that transitions t_4, t_{13}, t_{15}, and t_7 are uniquely covered by T-invariants T_1, T_2, T_3, and T_4, respectively. D_{sk} and D_{tk} are the service processing and transporting times under a T_k operation.

7. Average System Utilization
   (SU): $\mathrm{SU} = \mathrm{Prob}[m(p_{s1}) < S]$ (7)
8. Average Network
   Utilization (NU): $\mathrm{NU} = \mathrm{Prob}[m(\mathrm{channel}) < C]$ (8)
9. Average System Starvation
   (SS): $\mathrm{SS} = \mathrm{Prob}[m(p_{s1} > 0) \wedge m(Q_1) = 0]$ (9)
10. Percentage of Cancel Error
    Occurring (δ): $\delta = \dfrac{\alpha_3}{1 - \alpha_1} \times 100\%$ (10)

D. Numerical Results and Discussion

Numerical Values for Firing Times: The performance and sensitivity analysis has been conducted around a nominal operating point. The following numerical values for the median delays of the timed transitions are based on experience gathered by the System Integration Group of the CIM program at RPI [10], with the speed of PC's of 1 MIPS and the speed of communication network of 1 MBPS (the parentheses give the ranges for the sensitivity analysis, and all times are in unit of ms): *EnDecode_time_v* = 6.0 (1.5–10.5), *Travel_time_v* = 4.0 (1.0–7.0): The length of the message to be transmitted is dependent on the type of request (or response) made by the requester (or responder), thus, these delays are variable. *EnDecode_time_c* = 1.5, *Travel_time_c* = 1.0: The lengths of the control massages are constant, therefore these delays are constant. *Error_detection_time* = 2× *Endecode_time_c*: The delay caused by firing *t_15*. In this case, the canceling request message has to be first decoded, and afterwards an error massage encoded. *Arrival_time* = 100 (50–150), *Exec_time* = 50 (25–75) are network independent and depend only on the application at hand. It has been assumed that a canceling request was issued one out of 20 times, and that 99% of the canceling request will be accepted if the service has not been finished. We further assumed that only one communication channel is available and limited the number of outstanding services to be one ($C = S = 1$).

Numerical Results and Discussion: The plots are labeled in the following way: **AR**–arrival time is being varied; **EX**–execution time being varied; **EN**–encoding/Decoding time being varied; **TR**–traveling time being varied. Finally, the solid line indicates $K = 1$, the dashed line $K = 5$, the dotted line $K = 10$, and the dot-dashed $K = 20$.

Fig. 8 shows the number of waiting services increases rapidly as the interarrival time drops to 80 ms. Places Q_1, Q_2

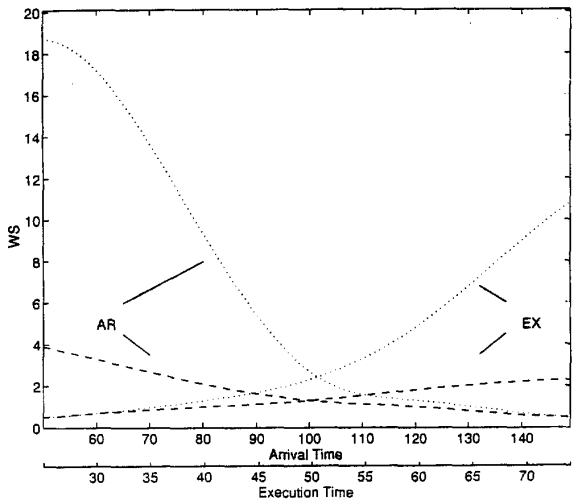


Fig. 8. Average number of waiting services versus arriving and execution times.

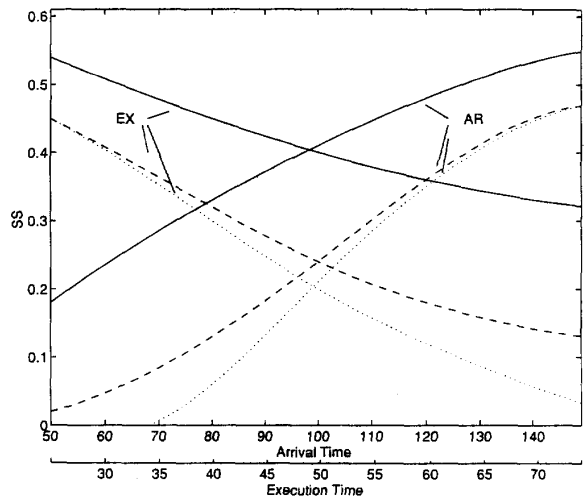


Fig. 9. Average system starvation versus arriving and execution times.

and transitions t_q, t_1 can be considered as a $M/G/1/K$ queuing system. For $M/G/1/\infty$ queue (large K), this implies that the traffic intensity might become larger than one, resulting in an infinite queue length. In terms of MMS, this means that service requests would have to be balked. The same effect can be observed for an increase in execution time, although the effect is not as severe. As expected, numerical results show the waiting time for service increases with queue size. Depending on the application, especially for real-time control systems, the waiting time might not be admissible. It also indicates that the average service processing time appears to change almost linearly. This is due to the fact that T_1 dominates the average service time. It is interesting to note that the arrival rate has no influence on processing time. From numerical results it is also obvious that the system is rather sensitive with regard to the arrival rate. MMS is rapidly driven into saturation as the arrival time decreases, but it is less sensitive to a change in the execution rate. Fig. 9 indicates that MMS is starved for a reasonable fraction of the time, so that an arbitrarily arriving customer can enter service immediately. At the operating point, the system is idle approximately 42% of the time. Finally, Fig. 10 gives the percentage of cancel error. As expected, the interarrival time has no influence on the error. On other hand the error decreases as the system becomes slower. At the first glance, this is counter-intuitive, but a look at T_3 confirms that a slow system implies a high reliability. As the time for decoding, traveling and execution increases, the probability that the task is completed before canceling the request arrives decreases.

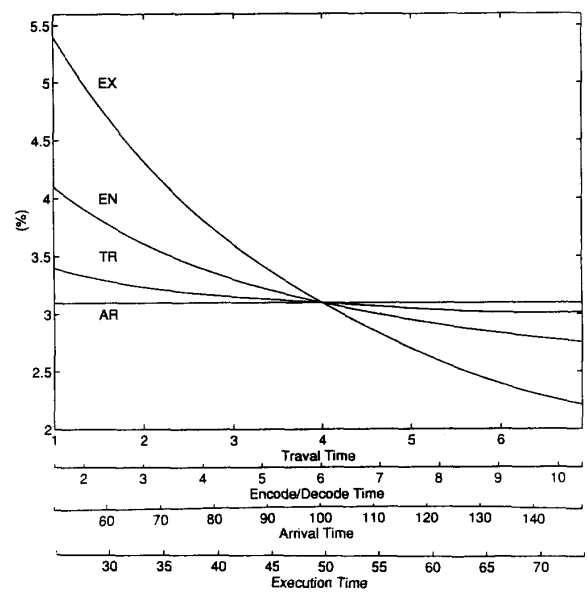


Fig. 10. Percentage of cancel error occurring versus four time parameters.

VI. Conclusion

This paper described a PN based method toward design and analysis of the MMS, a communication protocol for manufacturing environments. Starting with the service unit automata for individual MMS services, a procedure for constructing PN's to specify the MMS service connections have been developed. Using the execution rule of PN's, the service connection PN's describe all possible sequences of service primitives allowed during a service connection between MMS-providers and MMS-users. Analysis of service processes for protocol design has been conducted in terms of analytic concepts and methods developed in PN theory.

An advantage of describing MMS with PN's is that the model is also applicable for performance evaluation and sensitivity analysis. A generalized stochastic PN model has been constructed based on the confirmed services and a simplified computational scheme using T-invariant analysis has been proposed. The numerical results have shown that the service arriving rate has significant effect on several important per-

formance indices, such as the number of waiting services and system utilization. This indicates that MMS performance is quite sensitive with respect to the application environments. Thus, for real-time control systems with high communication intensity, the time delay through MMS might be unacceptable and some service requests may have to be balked. To solve the problem, one has to either reduce the intensity of message traffic or simplify the MMS communication protocol, or do both.

APPENDIX
INTERPRETATION OF THE GSPN MODEL

A. Interpretation of Places

Q_1	Queue of waiting services;
Q_2	Number of services allowed to enter Q_1;
p_{1s}	Service requester idle;
p_{2s}	Service responder idle;
p_1	Service pending at service requester;
p_2	No service cancel request is issued;
p_3	A service cancel request is issued;
p_4	Service cancel pending from requester;
p_5	Service pending at responder;
p_6	Service cancel pending from responder;
p_7	A service cancel request is refused;
p_8	A service cancel request is accepted; and
channel	Number of communication channels available between requester and responder.

B. Interpretation of Transitions

t_Q	Service has arrived;
t_1	Issuing a service request;
t_{2i}	Entering no service canceling state;
t_{3i}	Entering service canceling state;
t_4	Receiving a positive service response;
t_5	Issuing a service cancel request;
t_6	Receiving a negative response for cancel request;
t_7	Receiving a positive response for cancel request;
t_8	Receiving a service request;
t_9	Executing a request and issuing a response;
t_{10}	Receiving a service cancel request;
t_{11i}	Entering the refusing service cancel state;
t_{12i}	Entering the accepting service cancel state;
t_{13}	Issuing a negative response for cancel request;
t_{14}	Issuing a positive response for cancel request; and
t_{15}	Issuing a cancel error.

C. Average Firing Times of Transitions

The associate of firing times with transitions is:

Arriving_time	t_Q;
EnDecode_time_v	t_1 and t_8;
EnDecode_time_c	$t_4, t_5, t_6, t_7, t_{10}, t_{13}$, and t_{14};
Error_Detection_time	t_{15}
Exec_time	t_9
Travel_time_V	n_1 and n_2; and
Travel_time_c	n_3, n_4, and n_5.

D. Random Switches

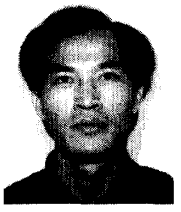
There are two switches, (t_{2i}, t_{3i}) and (t_{11i}, t_{12i}). Let $Prob_c$ and $Prob_a$ be the probabilities of issuing and accepting a cancel request, respectively. Thus, the probabilities of selecting t_2 and t_3 are 1-$Prob_c$ and $Prob_c$, and of selecting t_{11i} and t_{12i} are 1-$Prob_a$ and $Prob_a$, respectively.

ACKNOWLEDGMENT

The authors appreciate the supports from Dr. G. Ciardo by providing the SPNP package and Dr. G. Chiola by providing the GreatSPN package.

REFERENCES

[1] ISO/IS 7498, *Information Processing Systems-Open Systems Interconnection—Basic Reference Model*, ISO International Standard 7498, 1987.
[2] ISO/DIS 9506, *Manufacturing Message Specification*, ISO Draft International Standard 9506, Part One and Part Two, 1987.
[3] L. McGuffin, "MMS over MAP 2.1 specification," *ITI 1016*, Revision 1.0, 1987.
[4] H. J. Burkhardt, H. Eckert and P. Prinoth, "Modelling of OSI-communication services and protocols using predicate/transition nets," in *Proc. IFIP WG 6.1, 4th Int. Workshop on Protocol Specification, Testing and Verification*, Skytop Lodge, PA, 1984, pp. 165–192.
[5] J. Billington, "Specification of the transport service using numerical Petri nets," in *Proc. IFIP WG 6.1, 2nd Int. Workshop on Protocol Specification, Testing and Verification*, Idyllwild, CA, 1982, pp. 77–100.
[6] J. Billington, "Abstract specification of the ISO transport service definition using labeled numerical Petri nets," in *Proc. IFIP WG 6.1, 3rd Int. Workshop on Protocol Specification, Testing and Verification*, Ruschlikon, Switzerland, 1983, pp. 173–185.
[7] F.-Y. Wang and K. Gildea, "A colored Petri net model for connection management services in MMS," *Comput. Commun. Rev.*, vol. 19, no. 3, pp. 76–98, 1989.
[8] F.-Y. Wang, K. Gildea and A. Rubenstein, "A note on confirmed services in MMS," *CIM Program*, Center for Manufacturing Productivity and Technology Transfer, RPI, Troy, NY, DOC#CIMMN89TR181, 1989.
[9] F.-Y. Wang and K. Gildea, "Manufacturing message specification design and analysis using Petri nets," in *Proc. 1st Int. Workshop on Formal Methods in Eng. Design, Manufacturing, and Assembly*, Colorado Springs, CO, 1990, pp. 184–201.
[10] F.-Y. Wang, H. Jungnitz and K. Gildea, "Performance evaluation of MMS using generalized stochastic Petri nets," in *Proc. IEEE Int. Conf. on Robotics and Automat.*, Sacramento, CA, 1991, vol. 2, pp. 1573–1579.
[11] F.-Y. Wang and G. N. Saridis. *Coordination Theory of Intelligent Machines: Applications in CIM and Intelligent Robotic Systems*. Boston: Kluwer, 1994.
[12] Fei-Yue Wang and G. N. Saridis, "Task translation and integration specification in intelligent machines," *IEEE Trans. Robotics and Automat.*, vol. 9, no. 3, pp. 257–271, 1993.
[13] M. C. Zhou and F. DiCesare, *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*. Boston: Kluwer, 1992.
[14] M. C. Zhou and F. DiCesare, "Parallel and sequential mutual exclusions for Petri net modeling of manufacturing systems with shared resources," *IEEE Trans. Robotics and Automat.*, vol. 7, no. 4, 1991.
[15] K. Jensen, "Coloured Petri Nets and the invariant method," *Theoretical Comput. Sci.*, vol. 14, pp. 317–376, 1981.
[16] J. Peterson, *Petri Net Theory and the Modeling of Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1981.
[17] M. Molly, "Performance analysis using stochastic Petri nets," *IEEE Trans. Comput.*, vol. 31, no. 9, pp. 913–917, 1982.
[18] G. Ciardo and J. K. Muppala, *Manual for the SPNP Package*,, Version 3.0, Department of Computer Science, Duke University, 1990.
[19] G. Chiola, *GreatSPN User's Manual*, Dipartamento di Informatica, Universita degli Studi di Torino, Corso Svizzera 185, 10149 Torino, Italy, 1990.



Fei-Yue Wang (S'89-M'90) received the Ph.D. degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, in 1990.

From 1988 to 1990 he was with the NASA Center for Intelligent Robotic Systems for Space Exploration at Rensselaer Polytechnic Institute. He joined the University of Arizona, Tucson, in 1990 as an Assistant Professor of systems and industrial engineering. He is the co-author of the book *Coordination Theory of Intelligent Machines: Applications in Intelligent Robotic Systems and CIM Systems*, and has published more than 40 refereed journal articles and book chapters, and about 50 conference papers and technical reports in the areas of applied mathematics, mechanics, mechatronics, artificial intelligence, control theory, communication, robotics and automation, computer-integrated manufacturing, fuzzy logic, neural networks, and intelligent machines. His fields of interest include mechatronics, robotics and automation, computer vision, computer-integrated manufacturing, intelligent controls and intelligent systems.

Dr. Wang is a member of Sigma Xi, Association for Computing Machinery (ACM), and National Council on Systems Engineering (NCoSE). He is an associate editor of *Intelligent Automation & Soft Computing* and the editor-in-chief of the *World Scientific Book* series on *Intelligent Control & Intelligent Automation*.

Kevin Gildea received the B.S. from the University of Scranton and the M.S. and Ph.D. in computer science from Rensselaer Polytechnic Institute, Troy, NY.

He has been working for IBM in the Large Scale Computing Division since 1982. From 1982 to 1987 he worked on the programming which controls various tools used in the manufacture of mainframe computers and on the programming which integrated those tools into an overall manufacturing control system. In 1987, he became IBM's Resident Engineer in Rensselaer Polytechnic Institute's Center for Manufacturing productivity where he worked on the Computer Integrated Manufacturing Program until 1992. He is currently working on the development of the communication subsystem for IBM's POWER Parallel Systems.

Hauke Jungnitz received the Ph.D. degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, in 1992.

From 1989 to 1992 he was a research assistant at the NASA Center for Intelligent Robotic Systems for Space Exploration at RPI. He joined Mercedes-Benz AG, Stuttgart, Germany, as a Senior Engineer in 1992. His major research area is performance analysis for manufacturing systems using Petri nets.

Deqian D. Chen received the M.S. degree in electrical engineering from Zhejiang University in 1988. He is currently a Ph.D. candidate in the Department of Systems and Industrial Engineering at the University of Arizona, Tucson, AZ. Since 1992, he has been working with Dr. Fei-Yue Wang in various problems in CIM, robotics and automation. His main area of research interests is intelligent controls based on fuzzy logic and neural networks for complex systems. Other research includes hardware and software designs for computer and electrical/electronic systems.