

On the supervisory control of distributed high-performance computing systems in engineering

H. Westphal, S. Menge

*Institute of Automation Technology, University of Bremen,
D-28359 Bremen, Germany*

Abstract

A logical structure for *understanding* and *control* of basic *performance considerations* of distributed high-performance computing systems in engineering is presented in Westphal *et al.* [1] and Westphal [2]. This structure is expanded in this paper on a methodology for the supervisory control of such systems. The supervisory control can be realized using the performance considerations mentioned above and the *generalized stochastic Petri net* (GSPN) model of the *fibre distributed data interface* (FDDI) presented in this paper.

1 Introduction

The growing requirements on manufacturing quality and the demands of customer-tailored products are the major concerns of plant managers and a strong motivation for using new modelling and control techniques and strategies for an intelligent information flow within the distributed high-performance computing systems. Here, the cost reduction and the operational efficiency of the *computer aided manufacturing* (CAM) is crucial for the survival and the future success of companies. To meet this, specific *computer aided engineering* (CAE) software tools have been developed and implemented to support the activities such as *computer aided design* (CAD) of new products, the *computer aided planning* (CAP) for the just-in-time production and the *computer assisted quality control* (CAQ) for optimal, standardized product quality. Distributed high-performance computer systems are needed because different applications demand different computing power, e.g. CAD stations need image processing power and in control appli-

100 High-Performance Computing in Engineering

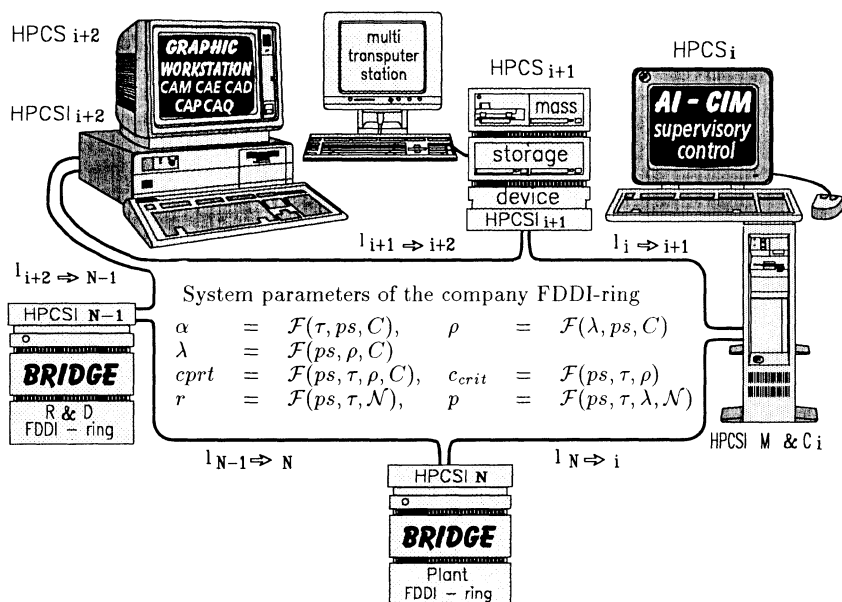


Figure 1: The company FDDI-ring and the system parameters.

cations real-time performance is decisive. In order to assure the economic use of such *computer aided components* (CAx), an *intelligent information flow*, databases, and mass storage devices have to be implemented and interconnected using a high-performance communication system.

Figure 1 shows the topology of a distributed *high-performance computing system* (HPCS) interconnected using FDDI *high-performance computing system interfaces* (HPCSI). The system parameters are derived in Westphal [2]. One HPCS is, preferably, to be interconnected with the communication system using a HPCSI equipped with a communications system *system parameter* measuring and control device (i.e. HPCSI M & C). In order to implement an supervisory control of the computing and communications systems this device should be able to receive, record, and transmit the system parameters of the communications system in real-time (Westphal [3]). Implementing further stochastic parameter optimization results in an optimal, adaptive control of the stochastic, time-discrete process that describes the performance and the behaviour of the *distributed computer control systems* (DCCS). The structure of a possible supervisory control of the system is shown in Figure 2.

In an intelligent information flow, on-line broadcasting of the DCCS performance parameters, like bandwidth, - and real-time response time parameters, as well as the future performance requests of the underlying communication system lead to an on-line, present and predictive *system knowledge*.

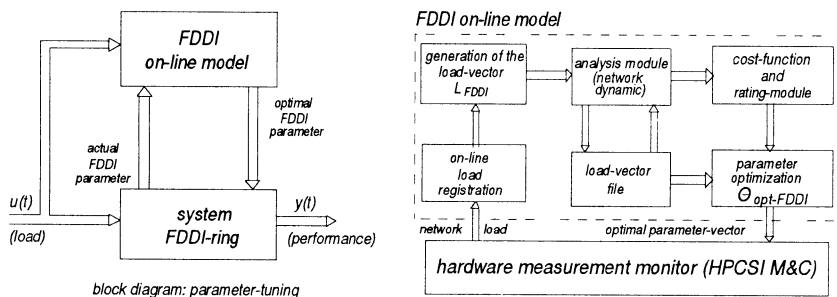


Figure 2: FDDI parameter tuning

Implementing an observer model, for instance a *generalized stochastic Petri net* (GSPN) model for a *supervisory control* of the *media access control* (MAC) protocol of the underlying communication system and merging the model with the dynamic system knowledge leads to a basic system model, that can be implemented within the DCCS and used to observe and control the entire communication system.

2 Advanced intelligent computer integrated manufacturing

To be able to implement a supervisory control of the high-performance communication system a reduced model in order to avoid modelling complexities is a prerequisite. This can be realized using the cell structure of a modern enterprise, which leads to the *hierarchical* modelling methodology of the *system parameter information flow*.

Figure 3 shows the hierarchical cell structure of an FDDI based production plant, as proposed here, where a monitor station collects the filtered and compressed data along the underlying rings (Westphal [4]). The hierarchical cell structure allows the construction of a *reduced model* of the system that avoids modelling complexities.

Figure 4 shows a FDDI backbone-ring of the company, a remote FDDI plant-ring, and a remote FDDI production-cell-ring exchanging data.

In the hierarchical-based modelling methodology, the development of *one* model of a communication system cell is sufficient for the optimal adaptive supervisory control of the enterprise-wide distributed high-performance computing and communication system. This is defined *advanced intelligent computer integrated manufacturing* (AI-CIM) (Westphal [4]).

The AI-CIM idea leads to an advanced, modular system for autonomous processes and for production plant automation, interconnected by an *hierarchical* communication infrastructure. This *plant within a plant* and *communications system within a communications systems* structure makes the high-performance communications infrastructure of an enterprise more practical and less expensive.

102 High-Performance Computing in Engineering

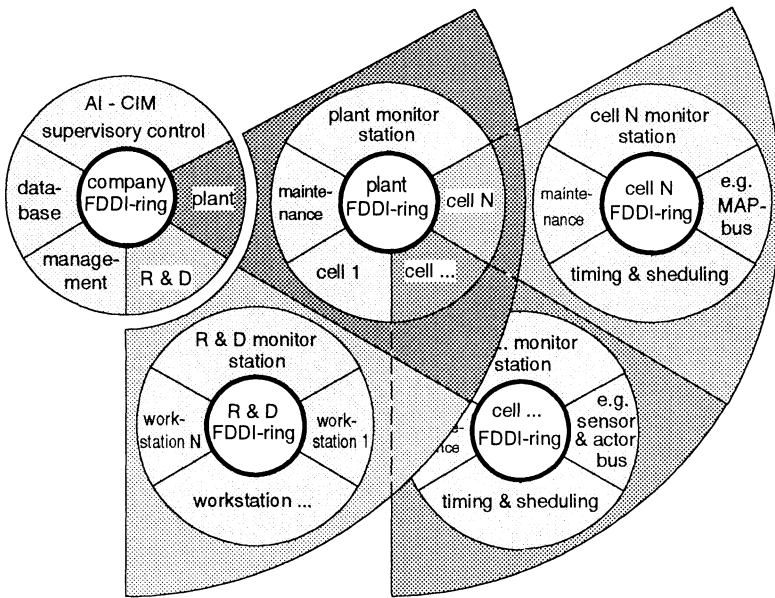


Figure 3: Hierarchical cell structure of an enterprise

The following chapter will cover the *generalized stochastic Petri net* (GSPN) model of an FDDI-ring, interconnecting the high-performance computers of an enterprise.

3 A GSPN approach to computer network performance analysis

The *generalized stochastic Petri net* (GSPN) model of an FDDI station is presented in Figure 5. Table 1 and Table 2 give an overview and an explanation of each transition and place of the GSPN.

In an FDDI ring, a token is circulating from an upstream station to a downstream station. The station, possessing the token can send FDDI data frames along the ring and after finishing the transmission, it passes the token to the next downstream station.

The token passed can generally arrive early or late at the next station (p1). When the token is early, synchronous and asynchronous frames can be sent (t2, p4, t4, p7, t8, p8, t9, p9, t10, p10). When there is no request for transmitting data, (p4, t6, p8, t16, p10) will be the covered place-transition sequence. The FDDI petri net model is basically composed of three main blocks: The *token rotation timer* (TRT), the *synchronous frame generator* and the *asynchronous frame generator*. When a token arrives at a station, it is registered as an early or late one and a mark is either on p3 or on p4.

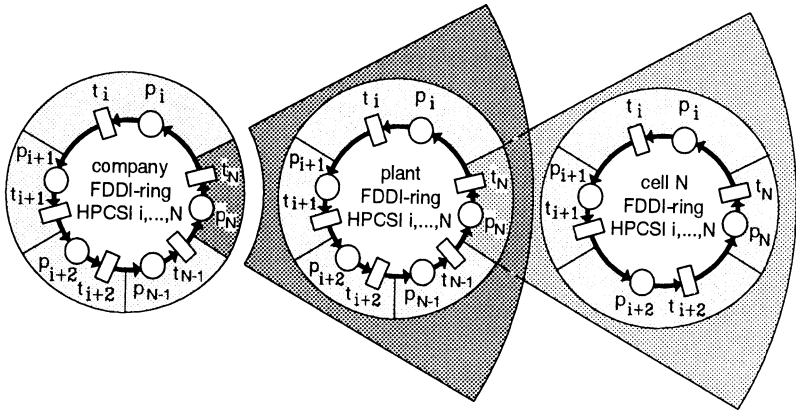


Figure 4: Hierarchical information flow of an enterprise

When synchronous frames are queued for transmission (mark on p_{13}) t_4 will fire, if not (mark on p_{14}) t_6 will fire. When a mark is on p_7 , the station starts to transmit synchronous frames. The delay related to transition t_8 represents the allocated synchronous bandwidth. When a mark is on p_8 , the synchronous transfer is terminated. When asynchronous frames are queued (mark on p_{16}), t_9 fires and asynchronous frames are transmitted. The related time to transition t_{10} is the average asynchronous transmission time. When a mark is on p_{10} , all transmission activities are terminated and the token is passed on the ring. The firing delays of the transitions t_8 , t_{10} , t_{12} , t_{13} and t_{15} are exponentially distributed.

4 GSPNs and Markov chains

A *generalized stochastic Petri net* (GSPN) can be described using a finite set of places P (circles), transitions T (rectangles), bows B (arrows), tokens (black circles), and the starting condition M_0 :

$$PN = (P, T, B, M_0) \quad (1)$$

$$P = (p_1, p_2, \dots, p_n) \quad (2)$$

$$T = (t_1, t_2, \dots, t_m) \quad (3)$$

$$B \subseteq (P \times T) \cup (T \times P) \quad (4)$$

$$M_0 = (m_1, m_2, \dots, m_n) \quad (5)$$

To implement time we define exponential distributed firing times λ_i for the transitions $t_i \in T$. The mean firing time $E_{(t_i)}$ is the reciprocal of the firing rate:

$$E_{(t_i)} = \lambda^{-1}. \quad (6)$$

104 High-Performance Computing in Engineering

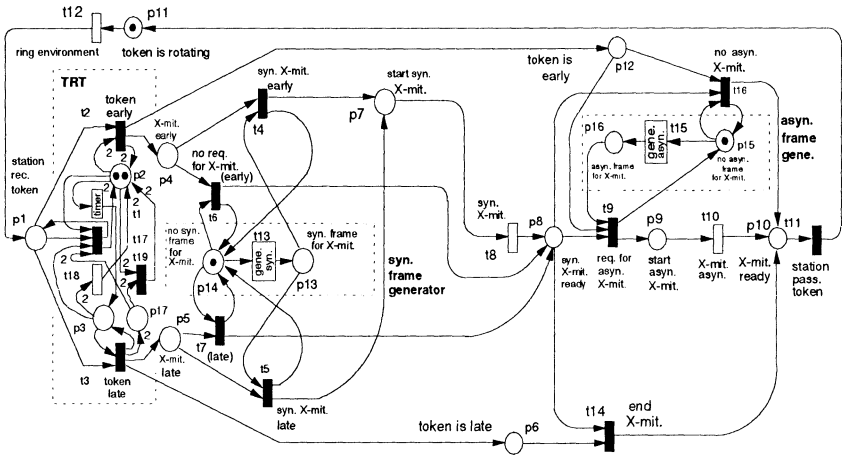


Figure 5: FDDI Petri Net

Every GSPN can be transformed in a time continuous Markov chain and every marking, reachable from the initial condition of the GSPN, is equal to a state of the Markov chain. The analysis of the GSPN is done by analyzing the reachability graph and we get the Markov chain of the underlying generator matrix $\underline{G}$ using the following equation:

$$S = S_v \cup S_t, \quad (7)$$

where S_v describes the vanishing states v , and S_t the tangible states t and we get $\underline{G}$:

$$\underline{G} = \begin{pmatrix} \underline{P}_t & \underline{P}_{t,v} \\ \underline{P}_{v,t} & \underline{P}_v \end{pmatrix}. \quad (8)$$

$\underline{P}_t$ and $\underline{P}_{t,v}$ describe the transition matrix. $\underline{P}_{t,v}$ describe the transition matrix from timed to nontimed states and we get:

$$\underline{T} = (\underline{I} - \underline{P}_v)^{-1}. \quad (9)$$

The matrix of the Markov chain $\underline{G}_R$ is:

$$\underline{G}_R = \underline{P}_t + \underline{P}_{t,v} \underline{T} \underline{P}_{v,t}. \quad (10)$$

$\underline{G}_R$ describes the generator matrix of the reduced Markov chain and we get the same results as using the origin matrix $\underline{G}$.

If a static state is available we get :

$$p_j = \lim_{t \rightarrow \infty} p_j(t) \quad (11)$$

$$\lim_{t \rightarrow \infty} \frac{d}{dt} p_j(t) = 0 \quad (12)$$

Table 1: FDDI GSPN transition table

Transitions		
name	acronym	meaning
t1	• TRT timer (timed)	• fires when TTRT expires
t2	• token early (immediate)	• fires when incoming token is early (TRT has not expired)
t3	• token late (immediate)	• fires when incoming token is late (TRT has expired)
t4	• syn. X-mit. early (immediate)	• fires when synchronous frames are queued for transmission and incoming token was early
t5	• syn. X-mit. late (immediate)	• fires when synchronous frames are queued for transmission and incoming token was late
t6	• no req. for X-mit. early (immediate)	• fires when synchronous frames are queued for transmission and incoming token was early
t7	• no req. for X-mit. late (immediate)	• fires when no synchronous frames are queued for transmission and token was late
t8	• syn. X-mit. (timed)	• fires when all synchronous frames are transmitted
t9	• req. for asyn. X-mit. (immediate)	• fires when asynchronous frames are queued for transmission and synchronous transmission accomplished
t10	• asyn. X-mit. (timed)	• fires when all asynchronous frames are transmitted (THT expired)
t11	• station pass. token (immediate)	• fires when transmission is accomplished
t12	• ring environment (timed)	• fires when a token attains the station (t12 models the rest of the ring)
t13	• gene. (syn.) (timed)	• fires when the average synchronous frame generation time expires
t14	• end X-mit. (immediate)	• fires when synchronous transmission is ready and if the token was late
t15	• gene. (asyn.) (timed)	• fires when the average asynchronous frame generation time expires
t16	• no asyn. X-mit. (immediate)	• fires when no asynchronous frames are queued for transmission
t17	• token early (immediate)	• token early after it was late
t18	• reconfiguration (immediate)	• $TRT_{max} \leq 2 TRT_{opr}$, ISO [5]
t19	• X-mit. flag (immediate)	• clear late X-mit. flag

Table 2: FDDI GSPN place table

Places		
name	acronym	meaning
p1	• station rec. token	• station receives a token
p2	• TRT start	• TRT timer start, decrement, and reset
p3	• TRT stop	• TRT-timer stop, late counter
p4	• X-mit. early	• the incoming token is early TRT has not expired
p5	• X-mit. late	• the incoming token is late TRT has expired
p6	• token is late	• registers if token was late
p7	• start syn. X-mit.	• station starts the transmission of synchronous frames
p8	• syn. X-mit. ready	• all synchronous frames are transmitted
p9	• start asyn. X-mit.	• station starts the transmission of asynchronous frames
p10	• X-mit. ready	• all frames (synchronous and asynchronous) are transmitted
p11	• token is rotating	• token is rotating on the ring
p12	• token is early	• registers if token was early
p13	• syn. frame for X-mit.	• synchronous frame generator has frames queued for transmission
p14	• no syn. frame for X-mit.	• asynchronous frame generator has no frames queued for transmission
p15	• no asyn. frame for X-mit.	• asynchronous frame generator has no frames queued for transmission
p16	• asyn. frame for X-mit.	• asynchronous frame generator has frames queued for transmission
p17	• late X-mit. flag	• registers if token is late

$$\sum_{i \in S} p_i g_{i,j} = 0 \quad (13)$$

$$\sum_{i \in S} p_i = 1 \quad (14)$$

$$\underline{p} \underline{G} = \underline{0} \quad (15)$$

If no static state is available we have to use the Chapman/Kolmogorov differential equation:

$$\underline{p}'(t) = \underline{p}(t) \underline{G} \quad , \quad \underline{p}(t_0) = \underline{p}_0 \quad (16)$$

$$\underline{p}(t) = \underline{p}_0 e^{\underline{G}t}. \quad (17)$$

Adding a control vector:

$$\underline{p}'(t) = \underline{p}(t) \underline{G} + \underline{b} \underline{u} \quad , \quad \underline{p}(t_0) = \underline{p}_0, \quad (18)$$

leads to the control of the system and optimizing the system behaviour can be done by using the generator matrix $\underline{G}_{opt}$. $\underline{G}_{opt}$ can be found using an

106 High-Performance Computing in Engineering

optimal systems trajectory for the parameter functions. The optimal state trajectory is defined as:

$$p_{opt} \underline{G}_{op} = \underline{0}. \quad (19)$$

5 Conclusion

The paper presented a methodology for supervisory control of distributed high-performance computing systems in engineering that can be implemented using the system parameters derived in Westphal *et al.* [1], Westphal [2], and the methodology shown in this paper. Useful basic informations on the modelling and supervisory control methodology are given to the system analyst and designer, who will be able to select an adequate network and an optimum network controller for real-time applications. Quantitative performance results can be used as a support for the general design of modern hierarchical-based high-performance computing and communication systems. The presented analytical techniques permit the system engineers to model and analyze high-performance systems and to obtain approximations of the network performance. This puts also the development and maintenance of high-performance real-time communication systems on an analytic, engineering basis, making these systems easier to develop and maintain.

References

- [1] Westphal, H. and Popović, D. Performance evaluation of distributed transputer networks using high performance communication links. In C.A. Brebbia and H. Power, editors, *Applications of Supercomputers in Engineering, ASE 93*, Third international conference on ASE, pages 425 – 437, Bath, UK, 27 - 29 September 1993. Wessex Institute of Technology, WIT, Computational Mechanics Publications, Southampton Boston, Elsevier Applied Science, London New York.
- [2] Westphal, H. *Introduction into fundamentals of Distributed, High-Performance Computing Systems in Engineering*, volume 1 of *High-performance computing in Engineering: Introduction and Algorithm, Power, H. and Brebbia, C.A. (Eds.)*, chapter 1, pages 1 – 40. Computational Mechanics Publications, Southampton Boston, 1995.
- [3] Westphal, H. Talk on: Implementation and test of a low-cost FDDI ring with system parameter adaption for real-time process automation. University of Bremen, © Institute of Automation Technology, June 1992.
- [4] Westphal, H. and Popović, D. and Spaus, J.M. The role of FDDI in CAD/CAM Systems. In *10th IFAC/ISPE/OCRI International Conference on CAD/CAM, Robotics and Factories of the Future, CARs & FOF'94*, Ottawa, Canada, 21 - 24 August 1994. ISPE/IFAC.
- [5] ISO 9314-2. *Information processing systems - Fibre Distributed Data Interface (FDDI) - Part 2: Token Ring Media Access Control (MAC)*, 1989.