'92 KACC 1992. 10. 19~21

Scheduling Algorithm of Data Sampling Times in the Real-Time Distributed Control Systems

Seung Ho Hong

Department of Control and Instrumentation Engineering
Hanyang University

Abstract

The Real-time Distributed Control Systems (RDCS) consist of several distributed control processes which share a network medium to exchange their data. Performance of feedback control loops in the RDCS is subject to the network-induced delays from sensor to controller and from controller to actuator. The network-induced delays are directly dependent upon the data sampling times of the control components which share a network medium. In this study, a scheduling algorithm of determining data sampling times is developed using the *window concept*, where the sampling data from the control components dynamically share a limited number of windows.

I. Introduction

A large and complex control system is usually decomposed into several distributed single-function subsystems. The major advantages of decomposition include flexibility of operation, evolutionary design process, and ease of maintenance, diagnostics and monitoring. The sensor data and the process information generated from a variety of distributed subsystems are exchanged through the network. Networking also provides backup facilities in real-time applicaitons, and sharing of expensive and scarce resources. Thus, networks form the backbone of Real-time Distributed Control Systems (RDCS) where several complex dynamical processes are integrated into one large and complex system. Examples of RDCS include advanced aircraft and spacecraft, automated factories, chemical plants, and electric and nuclear power plants [1-3].

Interconnection of distributed components through networking increases reliability, flexibility, and user friendlyness in design, operation and management of the complex systems. However, performance of the real-time control systems could be adversely affected by the data delays induced by the network traffic [4]. Thus, RDCS must be designed such that network-induced delays do not exceed the pre-determined limitation. Network-induced delays are dependent on the intensity of the network traffic, which is directly dependent upon the data sampling time (or data interarrival time from the view point of the network system) of each component of the control system which share a network medium. Data sampling times of the control loops in the RDCS should be selected such that they:

- o satisfy the performance requirements of control system
- o maximize the utilization of network resources

The problem of time-constrained communication has been addressed in the literature [5, 6]. However, most of the previous works assumed the Poisson process of data arrival and infinite queue length, which cannot be directly applicable to the RDCS where each control component samples its data on a *periodic* basis, and queue capacity is usually restricted to *one* (this is because the control components in the RDCS always require most recently generated sensor and controller data). The objective of this study is to develop a scheduling algorithm which determines the data sampling times in the RDCS.

II. Real-time Distributed Control Systems (RDCS)

In the RDCS, a number of control loops share one network medium. Figure 1 shows an example of RDCS, where each control loop consists of a set of sensor, controller

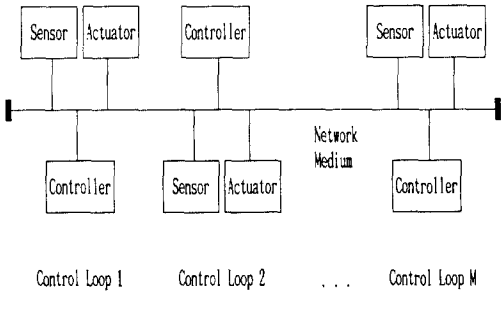
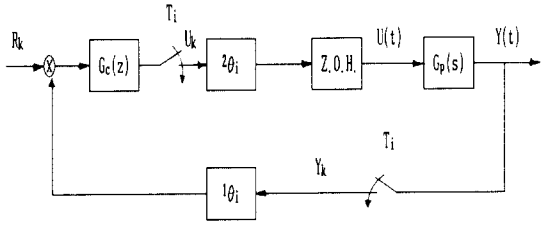


Figure 1. Real-time Distributed Control Systems

and actuator. In most industrial network systems, Medium Access Control (MAC) can be largely devided into two categories of (1) centralized controlled access of polling system (e.g. MIL-STD-1553B, FIP, etc.), and (2) distributed controlled access of token passing systems (e.g. IEEE 802.4 token bus, PROWAY C, etc.) [1]. Both polling and token passing systems are operated on the basis of cyclic service discipline, where data in the transmitter queue of the control components are served in a cyclic order. The algorithm developed in this study can be applicable to the cyclic services of both token passing and polling systems.

In the RDCS, sampled data have to wait in the transmitter queue of sensor or controller node until the server (token or polling signal) arrives. This causes the network-induced delay. Figure 2 shows the network-induced delays in a feedback control loop i, where ${}^1\theta_i$ is sensor-controller delay and ${}^2\theta_i$ is controller-actuator delay. The simulation study in [4] shows that network-induced delay has a characteristic of randomly time-varying. Up to now, generally applicable technique for the analysis of control system with randomly time-varying delay has not yet been reported. Instead of analyzing the control system with randomly time-varying delay, the approach of RDCS design presented in this study is to determine appropiate data sampling time based on the given requirements of maximum allowable netwrok induced delays at each control component.

For a given RDCS network, network parameteres usually consist of (1) the number of nodes in the medium, (2) data length (or data transmission time), and (3) data interarrival time at each node. In the RDCS, sensor and controller are usually packetized to a fixed length. Thus, the designer of RDCS should determine data interarrival time (or data sampling time from the view point of control system) at each node in order to satisfies the performance requirements of control systems.



$G_c(z)$: Controller Transfer Function
$G_p(s)$: Plant Transfer Function
Z.O.H. : Zero-Order Holder
${}^1\theta_i$: Sensor-Controller Delay
${}^2\theta_i$: Controller-Actuator Delay
T_i : Sampling Time of i-th Control Loop
U_k : Sampled Controller Data
Y_k : Sampled Sensor Data

Figure 2. Network-induced Delay in the i-th Control Loop

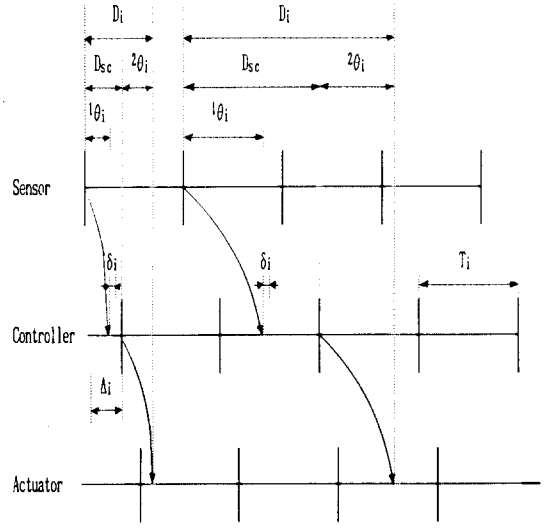


Figure 3. Loop Delays in the i-th Control Loop

Performance of a feedback control loop is directly dependent upon the loop delay [7]. The loop delay is measured from the instant when the sensor node samples a *tagged* sensor data to the instant when the controller data (which is generated based on the tagged sensor data) completely arrives at the actuator. For an identical sampling interval T_i of sensor, controller and actuator, total loop delay D_i of a control loop i is depicted in Figure 3. From the figure, total loop delay D_i is expressed as

$$D_i = D_{sc}(T_i, \Delta_i) + {}^2\theta_i \tag{2.1}$$

where

$$D_{sc}(T_i, \Delta_i) = \Delta_i + n\,T_i \tag{2.2}$$

and

$$n = \begin{cases} 0 & , \ {}^1\theta_i+\delta_i < \Delta_i \\ \mathrm{Int}[({}^1\theta_i+\delta_i-\Delta_i)/T_i]+1, & {}^1\theta_i+\delta_i \geq \Delta_i \end{cases} \tag{2.3}$$

where Int[x] implies integer part of x, and δ_i is the processing delay at the controller node. Δ_i is the sampling time skew between sensor and controller node in the loop i. In the control loop of RDCS, sensor and controller nodes are remotely located. The clocks at the sensor and controller nodes may not be synchronized due to their inaccuracy, which causes the sampling time skew. From (2.1) to (2.3), we see that the drift of Δ_i causes abrupt change of loop delay D_i. This sampling time skew can be bounded to a fixed value by periodically broadcasting a synchronization message from a designated node through the network medium. If the sampling time skew between the nodes can be bounded to the negligibly small value compared to the data sampling time ($\Delta_i \ll T_i$), and the controller processing time is also assumed to be negligibly small compared to the data sampling time ($\delta_i \ll T_i$), the loop delay is simplified as given in Figure 4, where the total loop delay is expressed as

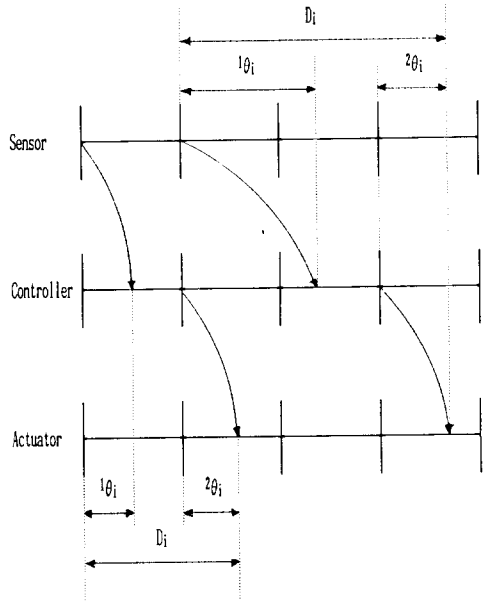


Figure 4. Simplified Loop Delays in the i-th Control Loop

$$D_i = n\,T_i + {}^{2}\theta_i \tag{2.4}$$

where

$$n = \mathrm{Int}[{}^{1}\theta_i / T_i] + 1 \tag{2.5}$$

III. Algorithm of Determining Data Sampling Times

Let's consider an RDCS which consists of M control loops. Loop i is composed of m_i nodes which transmit their data through medium (actuator node usually does not transmit its data). For a given set of RDCS parameters of (1) the number of nodes in the medium ($N = \sum_{i=1}^{M} m_i$), (2) packetized data length (L'), (3) data rate of network medium (B), and (4) maximum allowable delay at each control loop (Φ_i, i=1 to M), the objective is to determine the data sampling time T_i (i=1 to M) which satisfy the performance requirements of control systems, as well as considerably increase the utilization of network resources. The packetized data transmission time of all nodes is identical to L = L'/B, and the nodes in a control loop have identical data sampling time.

As mentioned earlier, the network-induced delay has a characteristic of randomly time-varying. Figure 5 shows that, when the network-induced delay happens to greater than the data sampling interval, more than one sensor data arrives during some controller sampling intervals, and only the last (most fresh) sensor data is used to generate controller singal. This causes *data rejection*. On the other hand, in some controller sampling interval, no sensor data arrives, which causes *vacant sampling*. Data rejection and vacant sampling not only degrades control performance but also introduces distortion of the controller signal. The distortion of control input causes

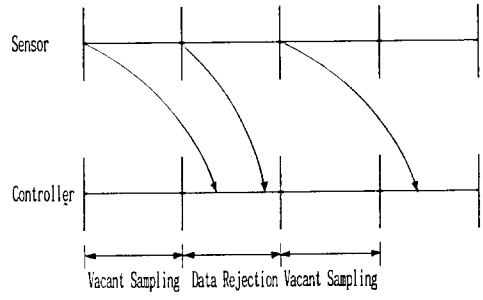


Figure 5. Illustration of Data Rejection and Vacant Sampling

high frequency noise in the actuator leading to excessive wear. Thus, every sensor (controller) data should be arrived at the controller (actuator) node before the next sensor (controller) data is sampled, i.e., for loop i which consists of m_i nodes,

$${}^{1}\theta_i < T_i,\ {}^{2}\theta_i < T_i,\ \ldots,\ {}^{m_i}\theta_i < T_i \tag{3.1}$$

where ${}^{j}\theta_i$ is the data latency of the node j in the loop i. On the other hand, the loop delay of a control loop i should not exceed its limitation Φ_i, i.e.,

$$D_i < m_i T_i \le \Phi_i \tag{3.2}$$

Deadline of a node j is defined as the time interval required for a data to be arrived at the receiver node before the next data is sampled. We say that *overflow* occurs if a data does not arrive at the receiver node within the deadline. A scheduling algorithm is *feasible* if the data sampling times are scheduled such that none of the nodes in the RDCS experiences overflow, as well as the loop delay of each control loop does not exceed its limitation.

Let T be a vector of the sampling times of M loops, i.e.,

$$T = [T_1,\ T_2,\ \ldots,\ T_M] \tag{3.3}$$

where T_i is the order of equal or increasing number, i.e., $T_i \le T_{i+1}$, $\forall i$. Let r_i be the maximum number of data that can be served by the server (token/polling signal) in the medium during the sampling interval of T_i, i.e.,

$$r_i = \mathrm{Int}[(T_i - N\sigma)/L] \tag{3.4}$$

where σ is overhead of the server at a node, which includes the server interpretation time and maximum server propagation delay ($\sigma \ll L$).

Lemma 1: If the total number of data served in the medium during each and every interval of T_j does not exceed r_j, node j will never experience overflow.

Proof: The worst case of data latency θ_j occurs when the node j releases the server immediately before it samples a data. The server will cyclically visit all the nodes in the medium. During this circulation, if (out of N nodes) only p nodes have waiting data when the server arrives,

then the data latency experienced by the node j will become $\theta_j = pL + N\sigma$. If p never exceeds r_j, then $\theta_j < T_j$ from (3.4), and node j will never experience overflow. ■

Let r be the maximum number of data that can be served in the medium during the smallest sampling time T_1, i.e.,

$$r = \text{Int}[(T_1 - N\sigma)/L] \tag{3.5}$$

As shown in Figure 6, T_1 consists of r wnidows. The length of each window is identical to the packet transmission time L.

Theorem 1: None of the nodes in the RDCS will experience overflow if no more than r data are served during each and every T_1.

Proof: Consider an arbitrary node j, where $T_j = k_j T_1$ ($k_j \geq 1$). If no more than r data are served during each and every T_1, then the number of data served during T_j will never exceed $r_j = k_j r$. From Lemma 1, the node j will never experience overflow ■

Remark 1: Theorem 1 conveys a fundamental *window concept* of the scheduling algorithm, i.e., during each and every T_1, N nodes in the medium dynamically share the r windows so that the data served during each and every T_1 does not exceed r. ■

The number of windows r in T_1 is determined from (3.2) and (3.5) as

$$r = \text{Int}[(\Phi_1/m_1 - N\sigma)/L] \tag{3.6}$$

Let's introduce a vector K of the ratio of sampling times with respect to the smallest sampling time T_1.

$$K = [k_1, k_2, \ldots, k_M],\ k_i = T_i/T_1,\ k_i \leq k_{i+1},\ \forall i \tag{3.7}$$

For a given K, let α_K be the average number of data sampled during T_1.

$$\alpha_K = \sum_{i=1}^{M} (m_i/k_i) \tag{3.8}$$

Lemma 2: All the data sampled during each and every T_M can be accommodated by the windows offered during T_M provided that (1) k_{i+1} is an integer multiple of k_i for all i, i.e., $k_i = 2^{n_i}$ where n_i is a positive integer including zero, and (2) $\alpha_K \leq r$ (T_M is the largest sampling time in the medium, and consists of k_M number of T_1 slots).

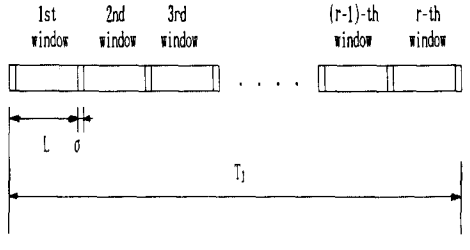


Figure 6. Illustration of Window Concept

Proof: Consider an arbitrary node j with data sampling time T_j, where $T_j \leq T_M$. If $k_j = 2^{n_j}$, the number of data generated from node j during each and every T_M is fixed to an *integer number* of k_M/k_j. Repeating this from j=1 to N, the total number generated from all nodes during each and every T_M is fixed to an integer number of $k_M \sum_{j=1}^{N} (1/k_j) = k_M \sum_{i=1}^{M} (m_i/k_i) = k_M \alpha_K$. Since the number of windows offered during T_M is $k_M r$, if $\alpha_K \leq r$, allthe data sampled during each and every T_M can be accommodated by the windows offered during T_M. ■

Remark 2: $\alpha_K > r$ implies that network capacity cannot accommodates input traffic, i.e., network system is overloaded. In this case, the RDCS designer must either use higher data rate (B) network, or reduce the number of nodes in the network. ■

From Lemma 2, the fact that fixed number of data are sampled during each and every interval of T_M leads to the fact that there exist a scheduling algorithm where less than of equal to r number of data are sampled during each and every interval of T_1 (see Theorem 1). The following Lemma conveys this idea.

Lemma 3: If $\alpha_K \leq r$ and $k_1 = 2^{n_1}$, then there exists a scheduling algorithm such that none of the node will experience overflow, i.e.,

$$S_j \leq r, \quad \forall\ j = 1 \text{ to } k_M \tag{3.9}$$

where S_j is the number of data sampled at the instant of A_j, and A_j (j=1 to k_M) is the beginning instant of j-th T_1 slot in the T_M. Figure 7 shows A_j in T_M.

Proof: T_j is an integer multiple of T_i for any j and i ($j \geq i$). Starting from the smallest sampling period, select r number of nodes with sampling periods of $T_1, T_2, \ldots, T_r$, and let them sample their data at the instant of A_1. These nodes will sample their data simultaneously at the inatants of $A_1 + nT_r$ (n=1, 2, ... , k_M/k_r) in the T_M. Since $\alpha_K \leq r$, all the data generated during each and every T_M is accommodated by the windows offered during T_M (see Lemma 2), which implies that these exists some instants between A_1 and $A_1 + T_r$, where less than r data are sampled. Pick one of the instants, which is the closest to A_1, and denote it as A_k. The number of data sampled at each $A_k + nT_r$ (n=1, 2, ... , k_M/k_r) instant will also be identical and less than r. Let the (r+1)-th node sample

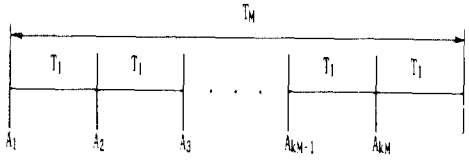


Figure 7. Illustration of A_j in T_M

its data at A_k, where $T_{r+1} \geq T_r$. (r+1)-th node will sample its data at the instants of A_k+nT_{r+1} (n=1, 2, ... ,k_M/k_{r+1}). Since T_{r+1} is an integer multiple of T_r, the number of data sampled at those instants still does not exceed r. Repeat this procedure to the (r+m)-th node (m=2, 3, ...) until r data are sampled at A_k. Then, find next instant which is the closest to A_k and less than r data are sampled, and repeat the procedure mentioned above upto the N-th node samples their data. This procedure leads to $S_j \leq r$, $\forall j$=1 to k_M. Since T_M is an integer multiple of T_i (i=1 to N), if an arbitrary node i samples its data at A_j of the first T_M, it will always sample its data at A_j of T_M afterward. Repeating this from node 1 to N, the number of data sampled at A_j shall be fixed to S_j for each and every T_M. ■

Based on the Lemma 2 and 3, the scheduling algorithm is summerized as the following theorem:

Theorem 2: The scheduling algorithm is feasible if k_i, i=1 to M, is selected such that:

$$(1)\ k_i = \left\| \frac{\Phi_i/m_i}{\Phi_1/m_1} \right\| \qquad (3.10)$$

$$(2)\ \alpha_K \leq r \qquad (3.11)$$

$$(3)\ S_j \leq r,\quad \forall\ j=1 \text{ to } k_M \qquad (3.12)$$

where $y=\|x\|$ implies y is one of 2^{n_i} (n_i is a positive integer including zero), which is the "closest" to but not exceed x.

Proof: Condition (1) implies that $k_i=2^{n_i}$, and at the same time the loop delay D_i of the control loop i does not exceed its limitation Φ_i (see 3.2). The conditions (2) and (3) follows from Lemma 2 and Lemma 3. ■

Remark 3: Since the data trasmission time at each node is L, condition (3) of the Theorem 2 (simultaneous sampling of several distributed nodes at A_i) can be relaxed to the condition that the maximum sampling time skew among these nodes is bounded to the packet transmission time L. As mentioned earlier, this can be easily accomplished by periodically transmitting the synchronization message from a designated node. ■

In case when the network traffic is lightly loaded such that $r \geq N$, the following simpler scheduling algorithm can be applied.

Theorem 3: If $r \geq N$, then the scheduling algorithm is feasible provided that T_i, i=1 to M, is selected such that

$$T_i = \Phi_i/m_i \qquad (3.13)$$

Proof: Since $T_j \geq T_1$, (j=1 to N), at most one data is sampled from node j during any interval of T_1. Thus, the maximum number of data sampled during T_1 is N. If r (the number of windows provided during T_1) is greater than N, no data will experience overflow during each and every T_1 (see Theorem 1). $T_i = \Phi_i/m_i$ satisfies the performance requirement of control system, and the scheduling algorithm is feasible. ■

Remark 4: Let U be the network utilization, which is defined as the fraction of time during which the network medium remains busy for data transmission. U is expressed as

$$U = \sum_{j=1}^{N} (L/T_i) = (m_iL/T_1) \sum_{i=1}^{M} (1/k_i) \qquad (3.14)$$

(3.14) implies that the utilization of network resource can be increased by proper choice of k_i. However increment of k_i is restricted by Φ_i, as given in (3.10). ■

Let t_j be the first data sampling instant of the node j within the first T_M interval. Based on the theorem 2 and 3, the scheduling algorithm of data sampling times in the RDCS can be summerized as follows:

Algorithm

Given: N, L, σ, Φ_i, m_i (i=1 to M)

Step 1: Determine T_1 and r as

$$T_1 = \min_i [\Phi_i/m_i]$$

$$r = \text{Int}\ [(T_1-N\sigma)/L]$$

Step 2: Determine the data sampling time at each control loop.

If $r \geq N$ (light traffic), then

$T_i = \Phi_i/m_i$, $\forall$ i=1 to M

Else

$$k_i = \left\| \frac{\Phi_i/m_i}{\Phi_1/m_1} \right\|, \ \forall\ i=1 \text{ to } M$$

$$\alpha_K = \sum_{i=1}^{M} (m_i/k_i)$$

If $\alpha_K > r$, then

(network is overloaded → reduce the number of nodes in the medium)

Else

$T_i = k_iT_1$

Step 3: Determine the data sampling instants within the first T_M.

For (i=1, j=1; $i \leq k_M$, $j \leq N$; i=i+1, j=j+1)

Do $t_j \leftarrow A_i$

while ($S_i \leq r$)

End

IV. Example of RDCS design

This section presents an example of RDCS design. The RDCS is assumed to be consisted of 5 control loops with 10 data transmitting nodes (5 controllers and 5 plants). Nodes (1, 2), (3, 4), (5, 6), (7, 8), (9, 10) are included in the control loops 1, 2, 3, 4, 5, respectively. Packetized data transmission time (L) and server overhead (σ) are 2 msec and 0.1 msec, respectively, and the maximum allowable loop delays at

each control loop are given by

$$[\Phi_1, \Phi_2, \Phi_3, \Phi_4, \Phi_5] = [20\ \text{msec}, 60\ \text{msec}, 100\ \text{msec}, 200\ \text{msec}, 400\ \text{msec}]$$

For a given RDCS, T_1 and r are determined from the Step 1 of the algorithm as 10 msec and 4, respectively. From Step 2, a vector K is determined as

$$[k_1, k_2, k_3, k_4, k_5] = [1, 2, 4, 8, 16]$$

For a given K, α_K is determined as 3.875 ($\alpha_K < r$), and the vector of the sampling times T is determined as

$$[T_1, T_2, T_3, T_4, T_5] = [10\ \text{msec}, 20\ \text{msec}, 40\ \text{msec}, 80\ \text{msec}, 160\ \text{msec}]$$

From Step 3, a vector of the first data sampling instants of node 1 to 10 is determined as

$$[t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}] = [0\ \text{msec}, 0\ \text{msec}, 0\ \text{msec}, 0\ \text{msec}, 10\ \text{msec}, 10\ \text{msec}, 30\ \text{msec}, 30\ \text{msec}, 70\ \text{msec}, 70\ \text{msec}]$$

Figure 8 depicts the data sampling instants of 10 nodes at the first T_M interval, where ①, ②, ... , ⑨, ⑩ represent the data sampling instants at the node 1, 2, ... , 9, 10, respectively. Note that no more than $r(=4)$ data are sampled at each and every slot of T_1. Exactly same data sampling pattern as shown in Figure 8 will be repeated for all intervals of T_M afterward. From (3.14), the network utilization U for a given example is 77.5% (unused portion of network resource includes indispensable bandwidth waste due to overhead of the server, σ). System utilization U_s is defined as the ratio of the number of windows used during T_M to the number of windows offered during T_M, i.e., $U_s = \alpha_K/r$. For a given example, U_s is estimated as 96.875%.

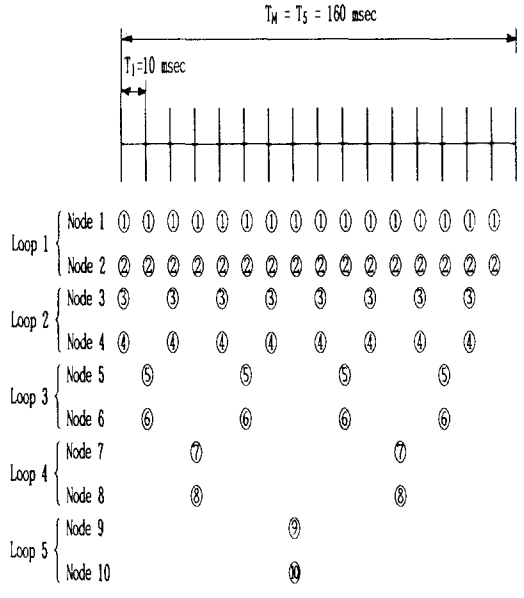


Figure 8. Example of RDCS Design

V. Conclusions

Performance of the feedback control loops in the RDCS is directly dependent upon the loop delay. Time-varying characteristics of the network-induced delay also causes data rejection/vacant sampling. Determination of data sampling times of the control components is very important when designing RDCS. The algorithm proposed in this study satisfies the fundamental requirements of RDCS design such as (1) limiting the loop delay within the maximum allowable value, (2) eliminating data rejection/vacant sampling, and (3) increasing the utilization of network resources. Due to the real-time performance requirement of control systems, conventional RDCS networks have been designed to limit the number of nodes accommodated by the network medium (e.g., $N < r$, see Theorem 3), which causes the waste of network resources. The scheduling algorithm introduced in this study considerably increase the utilization of network resources as well as satisfies the performance requirement of each control system in the RDCS.

References

1. IEEE Network Magazine: Special Issue on Communication for Manufacturing, Vol.2, No.3, May 1988.
2. A. Ray, Y. Halevi, S. Lee, S. H. Hong, "Network Access Protocols for Real-Time Distributed Digital Control Systems," IEEE Industrial Application Society Annual Meeting, Denver, CO, September 1986.
3. J. W. Meyer, "SAE AE-9B Draft Standard High Speed Token Passing Data Bus for Avionic Applications," IEEE/AIAA 7-th Digital Avionic Systems Conference, Fort Worth, TX, pp. 234-241, October 1986.
4. A. Ray, S. H. Hong, S. Lee and P. Egbelu, "Discrete-Event/Continuous-Time Simulation of Distributed Data Communication and Control Systems," Trans. of the Society for Computer Siumlation, Vol.5, No.1, pp. 71-85, January 1988.
5. J. F. Kurose, M. Schwartz and Y. Yemini, "Controlling Window Protocols for Time-Constrained Communication in Multiple Access Networks,: IEEE Transactions on Communications, Vol.36, No.1, pp. 41-49, January 1988.
6. K. W. Rose and B. Chen, "Optimal Scheduling of Interactive and Noninteractive Traffic in Telecommunication Systems," IEEE Transactions on Automatic Control, Vol.33, No.3, pp.261-276, March 1988.
7. A. Ray and Y. Halevi, "Integrated Communication and Control Systems: Part I-Analysis and PartII-Design Considerations," ASME Journal of Dynamic Systems, Measurement and Control, December 1988.