DECENTRALIZED PRIORITY CONTROL
IN DATA COMMUNICATION

by

L. Nisnevich

Technical Report No. 30
January 1974
\textbf{ABSTRACT}

This paper describes a new principle for the control of data transmission. Control is effected by a number of identical units which are uniformly distributed among sender-receivers. When a data transmission channel is not engaged, these units try to capture the channel. The unit having the highest address value captures the channel. We describe a procedure for assigning and changing unit priorities under the constraint that customer service indices remain below given levels. The suggested procedure can be used to assign priorities in queueing systems.
1. Introduction

In this paper, we are concerned with real-time parallel systems and in particular with data communication systems.

In parallel systems decentralized control increases flexibility and reliability. The following comment [1] deals with this important feature of parallel system organization.

"While graceful degradation is desirable in many commercial applications (e.g. process control, time sharing systems, etc.), it is essential in any military system where the results of complete system failure for even a short period of time could be catastrophic. The importance of this aspect of parallel processor techniques is further enhanced in military systems, since they face the loss of part of the system hardware not only from normal equipment and hardware failures, but also from the results of enemy action such as shell or bomb damage. In the latter case, however, it is important to note that this advantage of a parallel organization may be largely nullified unless the components of the parallel system are distributed physically as well as conceptually".

Recently, some publications have appeared [2,3,4] related to a data communication system with decentralized control proposal by Pierce [5]. Control is effected by distributing control units throughout the system. A Pierce system has many advantages, but data communication control in a real-time system may require interrupts and priority servicing. It is difficult to include such features in a Pierce system. Other proposals [6,7,8] for decentralized data communication priority control in real-time parallel computers have been based on the ideas of associative memory. Using these ideas, it is possible to achieve complete decentralization of priority control [13,14,15]. For such systems it is interesting to investigate the possibility of controlling priority parameters values in decentralized ways. This paper contains the outline of
some results relating to decentralized priority control. First we shall consider a scheme for changing priority values. We shall limit our discussion and consider only those systems in which the order of servicing is defined by absolute static priorities. The suggested methods may be extended to other systems.

Changing the order of servicing involves a reassignment of customer priorities. The purpose of such changes in this contest, is to find permissible priorities, i.e. to generate permissible values for multiple service indices. Such indices describe the level of customer service. Their values should not exceed the limits specified by the conditions under which customers operate. Examples of service indices are: average time between arrival and fulfillment of customer requirements for service, average number of customer requirements awaiting service more than a fixed limit, etc.

Such problems are important in real-time computer systems. Their solution can have a strong influence on the structure of those hardware and software components which interaction among the subsystems.

2. Changing Priority Parameters

The general approach to the problem of selecting priority parameters is as follows.

Assume that a system serves some fixed number of customers denoted by \( i; i = 1, 2, \ldots, N \).

On the basis of some a priori considerations all the customers are divided into several groups which are assigned different priorities. Quantitative characteristics of system behaviour are then specified only for the whole system and for the selected groups. These characteristics determine the sort of necessary priorities. Since each group is generally considered to have a unique priority the search routines which select adequate priority sets must use priority permutations.
The same system in which customers are separated into \( n \) groups and each group has a different priority can be described in another way.

Let us assign each customer \( i \) (\( i = 1, 2, \ldots, N \)) a priority parameter \( P_i \) and let the value of \( P_i \) be equal to the priority of the group to which customer belongs. Therefore all the customers belonging to the same group will have identical parameter values, while the total number of different values will be \( n \). The queue discipline is determined by priority parameter \( P_i \) in the following manner. Customer \( i \) has a higher priority with regard to customer \( j \) if \( P_i < P_j \). Requirements of customers having identical values of priority parameters are serviced in the order "first come, first serve". Using this approach the value of each priority parameter \( P_i \) can be changed independent of the other priority parameters.

To illustrate, consider a given number \( n \) of priority groups dividing all the customers. Pick any customer \( i \) and let his priority parameter value change continuously from \( -\infty \) to \( +\infty \). Observe the changes taking place in the system.

At the outset \( i \) will represent a separate group having the highest priority. Then as the value of \( P_i \) becomes equal to the parameter value of the highest priority group this customer will be included in it. Subsequently he will constitute a separate group again having a lower priority than this first group but a higher priority than the rest. This process will continue until customer \( i \) forms a separate group with the lowest priority.

3. Priority Parameter Value Changing to get Adequate Service

The possibility of changing priorities independently allows a system to respond in a very simple way to customer dissatisfaction when the level of service raises above a permissible value. Under such conditions the values of the priority parameters for these dissatisfied customers should be decreased while leaving the rest unchanged.
In order to appreciate the need for demonstrating the validity of this simple procedure, consider a procedure utilizing the general approach for selecting priority parameters.

Choose the group of customers with the highest priority among all the groups whose level of service is inadequate. Interchange the priority of this group with that of the group having the next higher priority. Then continue this process of interchanging group priorities until the level of service is adequate for all of the groups in the system.

Such a process does not always lead to a satisfactory queueing discipline even when there is such a discipline. By way of illustration, select three groups of customers whose assigned priorities corresponding to their numbers (1,2,3). Assume that under such a queueing discipline the level of service is inadequate only for the third group. Also assume that if we interchange priorities of the second and the third group the level of service will be inadequate only for the second group. Continuing in this manner we will return to the initial state and the process will cycle.

If assigning the first group the lowest priority would in fact solve the problem, this process will not find a solution.

Let us return now to the process of independent parameter change. This process ensures a solution of the problem whenever service indices possess certain properties described below. This method also permits the assignment of customers to a specified number of priority levels within the constraint of adequate service providing such an assignment is possible at all.

For a fixed arrival and service pattern, the values of service indices depend only on the vector \( \bar{P} = (P_1, P_2, \ldots, P_n) \).

Denote the total set of service indices which have the property that each index
associated with exactly one customer by

\[ \phi_1(P), \phi_2(P), \ldots, \phi_K(P). \]

Some subset of these indices indicate the service level of the first customer, another subset that of the second, etc. Denote the permissible level of the \( \phi_k(P) \) index by \( C_k \). Let the components of vector \( \bar{P}^* \) be a permissible set of priority values for which the following system of inequalities holds:

\[ \phi_k(\bar{P}^*) \leq C_k \quad (k = 1, 2, \ldots, K). \]

If we want to find a permissible set of parameters for a system in which the number of priority levels does not exceed \( n \) we have to introduce an additional constraint i.e. that the components of the vector \( \bar{P}^* \) take no more than \( n \) different values.

Note that any vector \( \bar{P} \) whose components differ from the corresponding components of a vector \( \bar{P}^{(0)} \) by the same value defines the same queueing discipline. Therefore, the service indices \( \phi_k(\bar{P}) \) do not change along the straight line defined by:

\[
\begin{align*}
P_1 &= P_1^{(0)} + t \\
P_2 &= P_2^{(0)} + t \\
&\quad \ldots \\
P_N &= P_N^{(0)} + t
\end{align*}
\]

where \( \bar{P}^{(0)} \) is any fixed vector and \( -\infty < t < +\infty \).

Many service indices used in practice such as the examples cited earlier for the case of service by a single machine or a single channel have a so called monotonicity property which is explained as follows.

Let \( \phi(\bar{P}) \) be a service index of customer \( i \). If the vector \( \bar{P}^{(1)} \) is generated from the vector \( \bar{P}^{(2)} \) by increasing the value of any priority parameter excluding the \( i \)-th then \( \phi(\bar{P}^{(1)}) \leq \phi(\bar{P}^{(2)}) \).

For monotonic systems in which all service indices \( \phi_1(\bar{P}), \phi_2(\bar{P}), \ldots, \phi_K(\bar{P}) \)
possess the monotonicity property, the problem of finding a permissible set \( \vec{P}^* \) of priority values is solved by the process which we will call "simple descent" and denote by \( \Pi \).

The initial set of parameter values may be represented by any integer component vector \( \vec{P}^{(0)} \). If at some stage of the process \( \Pi \) we have determined a set of parameter values \( \vec{P} \) then the next stage will involve the following procedure.

Choose a service index \( \phi_k(\vec{P}) \) for which the inequality \( \phi_k(\vec{P}) \leq C_k \) does not hold for the vector found at the previous stage. Let \( \phi_k(\vec{P}) \) be the service index of customer \( i \) which means that this customer receives inadequate service for this index \( \phi_k \).

We will assume that inadequate service for one index means inadequate service in general. The value of parameter \( P_i \) in the vector \( \vec{P} \) is replaced by \( (P_i - 1) \) while the rest of the components do not change. In [9] it has been shown that the process \( \Pi \) terminates in a finite number of stages not exceeding \( N(N - 1) \) and results in a permissible set of priority parameter values provided such a set actually exists.

If we initially adopt the queue discipline "first come, first serve", for all the requirements of all customers (this discipline is specified by a vector \( \vec{P}^{(0)} \) with equal components) then the process \( \Pi \) will result in a set of priority parameter values in a number of stages less than \( N(N - 1)/2 \). This set of values determines an adequate service level for all the customers with a minimum possible number of priority levels. In case an adequate set of parameter values does not exist the process will result in a situation demonstrating this fact during the same number of stages.

Therefore, it is obvious that the process having an initial vector with equal components solves the problem of assigning customers to priority levels when the number of levels must be limited.
At any stage of the process \( \Pi \) it might turn out that the values of serve indices of several customers lie above corresponding permissible values. In such a case one can choose one of these indices and reduce the value of the priority parameter associated with the corresponding customer. It is possible to reduce the values of several priority parameters simultaneously. The set of parameter values generated by the process \( \Pi \) does not depend on this choice while it is determined by the initial vector \( \vec{P}(0) \).

It is important to stress that in order to realize the search process one does not need to know the values of the functions \( \phi_k(\vec{P}) \) \( (k = 1, 2, \ldots, K) \). The only essential information is whether these values lie below specified levels. In other words it is only necessary to know if each customer is getting adequate service when the set of priority values is given.

Example.

There are four customers in the system. Their requests arrive independently in conformity with a Poisson distribution. The customers are serviced by a single channel with exponential service times. The parameters of the corresponding distributions for each customer are:

\[
\begin{align*}
\lambda_1 &= 1/5 \\
\lambda_2 &= 1/20 \\
\lambda_3 &= 1/10 \\
\lambda_4 &= 1/20 \\
\mu_1 &= 1 \\
\mu_2 &= 1/4 \\
\mu_3 &= 1/2 \\
\mu_4 &= 1/4
\end{align*}
\]

The average total time (ATT) any requirement is in the system should not exceed 4 units for the first customer, 5 for the second, 16 for the third, and 30 for the fourth.

To find the values of the priority parameters satisfying these constraints, consider the operations of process \( \Pi \). At the outset assign priorities in the order of increasing the values of the given constraints (ATT), i.e. the initial vector \( \vec{P}(0) \) will be
equal to (1,2,3,4). Under this queueing discipline the average total times are 1.25 for the first customer, 8.75 for the second, 9.17 for the third, and 37.5 for the fourth.

As we can observe this apparently natural set of priorities does not satisfy customers 2 and 4.

At the first stage of process \( \Pi \) reduce the priority parameter values of customers 2 and 4. Then we obtains priority vector \( \vec{p}(1) = (1,1,3,3) \). Under this queueing discipline, the average total times are 2.07, 5.07, 14.33, 16.33.

Now only the second customer is not satisfied. By reducing his priority parameter value, we obtain the vector \( \vec{p}(2) = (1,0,3,3) \), which gives times equal to 3.33, 5, 14.33, 16.33 thereby satisfying all of the customers.

In this case partitioning the customers into three priority levels is sufficient and in addition is the only possible solution.

This method of changing parameter values in order to determine an adequate queueing discipline can be applied when the order of servicing depends on priority in other ways, i.e. in other classes of priority systems [10]. To ensure efficiency of the process \( \Pi \), it is sufficient that service indices remain monotonic functions of priority parameters. Other papers [11] are devoted to the study of such systems and contain proofs of the efficiency of the simple descent procedure.

For real-time systems, the process \( \Pi \) may serve as a means of introducing dynamic priorities. Specifically, such priority will adapt to changes in the system and modify priority parameters when the values of service indices approach their limits provided external conditions change sufficiently slowly [9,10].

4. Decentralized Control of Priority Parameters

The possibility of independently changing the priority parameters of each customer in order to obtain a satisfactory mode of system operation is necessary for the design
of completely decentralized dynamic control of customer servicing. In the case we are considering, a change in a customer priority parameter only depends on the level of service provided for the customer at the current values of the system priority parameters. Therefore each customer or an automaton controlling his service only has to know if the current value of the corresponding service index is within the permissible range.

In the systems of parameters we are considering a reduction in the value of one parameter is equivalent to an increase in the values of all the rest. This property can be used to design a simple finite automata system which controls priority parameters in a completely decentralized manner. In such a system each automaton must not only know that its customer is not yet getting adequate service with the system parameters having their current values but must know if any priority parameter has reached its lowest possible value.

Consider a system of finite automata which changes its state in accordance with the above process $\Pi$.

The environment in which these automata are operating is described by boolean environment parameters $\psi_0, \psi_1, \psi_2, \ldots, \psi_N$. The value of $\psi_i$ ($i = 1, 2, \ldots, N$) is zero if the customer $i$ is satisfied and equals 1 otherwise.

If automaton $i$ is in a state corresponding to the least possible value of its priority parameter and $\psi_i = 1$ then this automaton performs an action bringing $\psi_0 = 1$.

If none of the automata perform such an action in the current state, the value of the parameter $\psi_0$ equals 0.

Graphs illustrating the state changes for automaton $i$ are shown in Fig. 1. Each state of the automaton is designated by a corresponding value of the priority parameter. Automaton $i$ may be in states $1, 2, \ldots, P_i^+, \ldots, P_i^{+1}$.
When the parameter values are \((\psi_0 = 0, \psi_i = 0)\) and \((\psi_0 = 1, \psi_i = 1)\), automaton \(i\) does not change its state. If the set of values is \((\psi_0 = 0, \psi_i = 1)\) then automaton \(i\) changes its state from \(u\) to \((u - 1)\), provided \(u\) does not equal 1. If it does, the automaton does not change its state but sets the value of parameter \(\psi_0\) equal to 1 for the next state of the system. When the set of values is \((\psi_0 = 1, \psi_i = 0)\) automaton \(i\) changes its state from \(u\) to \(u + 1\) provided \(u\) does not equal \(P_i^+\). If \(P_i = P_i^+\), then automaton remains in this state.

In [11] it has been shown that such a control procedure ensures that a system of automata will satisfy all the customers if the service indices are monotonic and global satisfaction is possible. The same papers contain a proof showing that for single-channel queueing systems with interruption any characteristic of the form \(E(f(\xi_t(\bar{P})))\) or \(E(f(\eta_t(\bar{P})))\) is monotonic. (Here \(E\) is the symbol of expectation; \(f(x)\) is a nondecreasing function; \(\xi_t(\bar{P})\) – random variable defining the length of time a requirement, arriving at the moment \(t\), remains in the system, where the parameter vector is \(\bar{P}\) and the system is in a steady-state; and \(\eta_t(\bar{P})\) – random variable specifying the number of requirements which have not been completely serviced by the system and remain in it at the moment \(t\), where the parameter vector is \(\bar{P}\) and the system is in a steady-state.)
In a more practical model of the behaviour of a decentralized control system, one should take into consideration that the information delivered to an automaton regarding the level of customer satisfaction may be inaccurate. This inaccuracy may be due either to defects in measuring the values of service indices or to the nature of these indices. For example, an index may be specified as a probability and may only be evaluated as the results of a series of observations.

The systems' behaviour in such a model can be described as follows:

For each customer $i$ assume a single composite characteristic $\phi_i(P)$, a single constant $C_i$ representing the upper bound on the parameter values of $\phi_i(P)$, and a function $\gamma_i(x)$ representing the probability that $\psi_i = 0$ if $\phi_i(P) \leq C_i$ where $x = \phi_i(P) - C_i$ (which is the amount by which the value of the service index exceeds its upper bound). With probability $\gamma_i(x)$, automaton $i$ makes a decision that when the parameter vector is $P$, the given constraint is satisfied ($\psi_i = 0$), and with probability $(1 - \gamma_i(x))$ decides that the given constraint is violated ($\psi_i = 1$). After the decision is made, automaton $i$ changes its state $P_i$ in accordance with the values of parameters $\psi_0$ and $\psi_1$ as stated earlier.

It is natural to assume that functions $\gamma_i(x)$ are non-increasing and

$$0 \leq \gamma_i(x) < \frac{1}{2}, \text{ if } x > 0;$$

$$\frac{1}{2} < \gamma_i(x) \leq 1, \text{ if } x < 0.$$ 

The study of a system controlled by two such automata is described in [12].

The automata change their states at discrete moments of time either simultaneously, i.e. synchronously, or asynchronously.

In the asynchronous model, the probability of automaton actuation is specified for each automaton. At any discrete moment the automaton becomes active with this probability independent of its history and of the system state. It becomes passive
with the complementary probability, (i.e. does not change its state). When the automaton is active, it behaves as stated above in the model with inaccurate information. In other words it decides the value of its environment parameter \( \psi_1 \) and changes its state in accordance with the values of environment parameters \( \psi_0 \) and \( \psi_1 \).

It has been shown that in such a model the mode of control is stable and that the ratio of the total time during which the system gives adequate service to the total system time increases asymptotically as a function of the number of adequate states in the deterministic model where the actuation probabilities obey some natural constraints.

5. Channel Capturing Units (CCU)

Let us consider the application of the principles described above to the design of completely decentralized control of data communication using a single channel.

Let this channel connect some sender-receivers (SR). Initially, we shall describe the decentralized control of the transmission time distribution to sender-receivers where the addresses of the SR's are fixed. This method was suggested in [13].

![Diagram of Channel Capturing Units (CCU)](image)

The address of all SR's are binary numbers having an identical number of digits.
Each SR is provided with a channel capturing unit (CCU) (Fig. 2).

If it is necessary to transmit a message, SR \(_i\) switches CCU \(_i\) into the active state. While in this state, CCU \(_i\) is watching the state of the channel.

The channel may be in three states: "transmit 1", "transmit 0", and "no transmission". The "no transmission" signal causes all of the active CCU's to transmit the highest digit (left most) of their addresses into the channel. In other words, after a message has been transmitted and the channel turns into the "no transmission" state, the active CCU's start transmitting their addresses into the channel.

If at least one of the transmitted digits is one then the channel is in the state "transmit 1", while if all the transmitted highest digits are zero, the channel is in the state "transmit 0". CCU \(_i\) of SR \(_i\) compares the state of the channel with its own highest digit. If the channel is in the state "transmit 1" and CCU \(_i\) has transmitted a "0" then it switches itself off and awaits the next "no transmission" state. Otherwise, CCU \(_i\) remains connected to the channel.

Each CCU which remains connected to the channel transmits into the channel its second address digit and the channel turns into state of either "transmit 1" or "transmit 0". All of the CCU's which have sent their second digit behave as they did following the transmission of their first digit. In other words, if the channel is in the state "transmit 1" then all the CCU's which have sent the signal "0" are switched off and await the next "no transmission" state. This process is repeated until all the address digits have been transmitted.

After the last digit is transmitted only one SR remains connected to the channel (its CCU having captured the channel). The address of this SR is greater that the addresses of all of the other SR's that have been trying to occupy the channel during this period.

Example

Let the senders that require transmission time have the addresses 1001; 1101; 0111
1100, (as shown in Fig. 2). During the transmission of the highest digit the SR having the address 0111 will stop transmitting its address. The rest will start transmitting the second digit of their addresses. At this moment the SR having the address 1001 will stop its transmission. The remaining two SR's will transmit the third digits of their addresses which are 0. Therefore, SR's will continue to transmit their addresses. While the fourth digit is being transmitted, the SR having the address 1100 will stop transmitting. Thus, the channel will be occupied by the SR with address 1101.

After all of the digits of the address have been transmitted, the CCU of the sender occupying the channel transmits the destination address and having gotten an answer as to whether it is free either starts sending a message for its SR or switches off and after some time repeats its attempt to establish communication with the same receiver.

6. The Distributed Control of a Multiplex Channel for Real-Time Systems

Now let the address of each SR consists of three parts: \( A = p s a \).

Part \( p \) is an inverted binary value of the customer priority parameter. This part is determined by an automaton operating in the simple descent mode described earlier.

Part \( s \) is employed to provide the order of servicing "first come, first serve", among the SR's having equal priority values. This part of \( A \) is formed in the following manner:

Consider first the case when there is a timer in the system sending signal "1" to all the SR's at equal intervals. The initial value of \( s \) is 0 for all of the SR's. When an SR becomes active each unit sent by the timer is added to the value of its \( s \). The process is continued until a transmission is completed. After a
message is transmitted and the SR becomes passive, its $s$ returns to 0. Therefore, $s$ registers the waiting time of its SR. The maximum possible registered waiting time is determined by the number of digits in $s$. Consider the case when there is no timer in the system. In addition to the communication channel (main channel) and the auxiliary channel transmitting environment parameter $\psi_0$ to the automata associated with part $p$, let the system provided with an auxiliary channel $S$ in place of the timer channel (Fig. 3).

![Fig. 3]

The initial value of $s$ is 0 for all the SR's. When a requirement arrives and the CCU becomes active, it sends the signal "1" into the auxiliary channel $S$. From this moment each "1" which is entered into the auxiliary channel $S$ due to the actuation of a channel capturing unit is added to the value of $s$ for each CCU awaiting transmission time. This process is identical to that of the timer, the only difference being that in place of the timer the auxiliary channel $S$ is used and $s$ does not register the waiting time but the length of the queue. Among the senders belonging to the same priority group, the one whose waiting time for message transmission is the longest has the greatest value of $s$. 
And finally address part a is the number (personal address) of an SR. This part is necessary only because address part ps may have the same value for two SR's. Thus single line data communication system with decentralized distributed control has two auxiliary channels and a main channel to transmit both the coded addresses of the SR's and regular messages. The auxiliary channels can be combined into one by special coding of the time pulses and the discrete state pulses \( \psi_0 \) at the moments of priority parameter corrections.

In such a system, transmission time is granted to an SR with the highest priority parameter value. In case of ties an SR with the longest waiting time, is chosen. If ties still remain, the greatest personal address is serviced.

A multiplex channel with interrupt capabilities may be organized by dividing the main channel into two channels; one to transmit the addresses of the SR's and the other to transmit the regular messages (see Fig. 4).

In this case the SR's try to capture the message channel by sending their addresses along the address channel after a "no transmission" state has appeared in the address channel. The CCU occupying the address channel transmits into this channel the address of the required receiver and connects its sender to the message
channel. After this it creates a "no transmission" state in the address channel (i.e. switches off this channel) but remains active. When the "no transmission" state appears, the CCU's start to fight for the channel by trying to transmit addresses into the address channel. A CCU only disconnects its sender from the message channel when it has been preempted by another CCU. In this case, its transmission time is interrupted and another SR with higher priority connects to the message channel and transmits its message. When a sender has completed its message it causes its CCU to turn into the passive state and the CCU does not make any attempt to reference the address channel until it again becomes active.

Papers [14, 15] are devoted to the problems of broadening the functional possibilities of multichannel and multiloop data communication systems based on the principles discussed above. These papers also consider design methods for such systems.

7. Conclusion

The decentralized distributed control systems described above have a great deal of flexibility. In case such a system connects highly organized devices such as computer units, computer terminals, or computers, the address of an SR may be modified according to the transmitted message.

Notice that the execution of \( X \leftarrow A + B \) requires that the contents of either \( A \) or \( B \) be moved to the accumulator and that the contents of the other be added. The initial choice is arbitrary i.e. has no effect upon the result, and from the control of a process may be viewed as consisting of two parts: a critical and a non-critical part which controls all other choices. (i.e. the arbitrary ones).

The efficiency of data processing systems may be improved by delaying arbitrary choice until execution time. This might be implemented by using distributed control mechanisms for non-critical decisions.
At the lowest level of the system, control is exercised by standard channel capturing units (CCU), thus providing flexible system which adopts to changing conditions.
REFERENCES


