3. The Estimation of the Communication Complexity of Parallel Algorithms

As it has been already described in the previous sections, time delays, which occur in the process of data transmission for arranging the interaction of separately functioning processes, may be crucial for parallel computation efficiency. This section discusses the analysis of the information flows, which occur in the course of carrying out parallel algorithms. The section gives the general characteristics of the data transmission mechanisms, analyses the complexity of the basic data communication operations and considers the methods of logical presentation of network topology. More detailed information of all these issues may be found in Kumar (1994), Quinn (2004).

This Section has been written based essentially on the teaching materials given in Kumar, et al. (1994).

3.1. Overview of Data Transmission Mechanisms

3.1.1. Routing Algorithms

The routing algorithms define the route of data transmission from the sending processor to the processor, which should receive the message. The methods for solving the problem are the following:

– *the optimum ones,* which always define the shortest path for data transmission, and *non-optimum* routing algorithms;

– *deterministic and adaptive methods of choosing routes* (the adaptive algorithms define the route of data transmission depending on the available load of communication channels).

Among the widely used optimum algorithms, there is the class of *dimension-ordered routing methods*. The search for data transmission routes is carried out in the methods for each topology dimension of communication network in turn. Thus, for the two-dimensional grid this approach leads to such type of routing, which involves data transmission first in one direction (for instance, horizontally till the vertical line of processors, which contains the assigned processor, is reached), and then the data is transmitted along the other direction (the given scheme is known as *the XY-routing algorithm*).

For a hypercube the coordinate nested routing scheme may consist, for instance, in cyclic data transmission to the processor along the dimension which is determined by the first different bit position in numbers of the processor-receiver and a processor holding currently the transmitted message.

3.1.2. Data Transmission Methods

The time necessary for transmitting data between the processors defines the communication overhead of the the duration of parallel algorithm execution in a multiprocessor computer system. The basic set of parameters, which can help to evaluate the data transmission time, consists of the following values:

– **initializing time** (t_s) characterizes the duration of preparing the message for transmission, the search of the route in the network etc.

– **control data transmission time** (t_h) between two neighboring processors (i.e. the processors, connected by a physical data transmission channel); to control data we may refer the message header, the error detection data block etc.;

– **transmission time of one data byte** along a data transmission channel (t_b); the duration of this transmission is defined by the communication channel bandwidth.

There are two main communication methods among the most widely used data transmission techniques (see, for instance, Kumar (1994)). The first one is oriented at transmitting messages as indivisible information blocks (*store-and-forward routing* or *SFR*). In case of this approach the processor, which contains a message for transmission, gets all the amount of data ready for transmission, defines the processor, which should receive the data, and initializes the operation of data transmission. The processor, to which the message has been sent, first receives all the transmitted data and only then begins to send the received message further along the route. The time of data transmission t_{comm} for the method of transmitting the message of m bytes along the route of length l is defined by the expression

$$t_{comm} = t_s + (mt_s + t_h)l .$$

If the messages are long enough, the control data transmission time may be neglected, and the expression for data transmission time may be written in a simplified way:

$$t_{comm} = t_s + mt_h l .$$

The second communication method is based on presenting the transmitted messages as information blocks of smaller sizes (*packets*). Data transmission, as a result, may be reduced to packet communication. In case of this method (*cut-through routing* or *CTR*) the receiving processor may send the data further along the route immediately after receiving the current packet without waiting for the termination of the whole message data transmission. The data transmission time in case of packet communication method will be defined by the following expression:

$$t_{comm} = t_s + mt_b + t_b l .$$

If we compare the obtained expressions, it is possible to notice that in the majority of cases the packet communication leads to faster data transmission. Besides, this approach decreases the need for memory for storing the transmitted data. Different communication channels may be used for packet communication simultaneously. On the other hand, the implementation of the packet communication requires the development of more complex hardware and software. It may also increase the overhead expenses (initialization time and control data transmission time). Deadlocks may also occur in case of packet communication.

3.2. The Complexity Analysis of the Data Transmission Operations

Despite all the variety of the data transmission operations in case of parallel methods of solving complex time-consuming problems, certain procedures of network processor interactions may be referred to the basic communication operations. Such operations are either widely used in parallel computation practice or other message passing operations may be reduced to them. It is also important that there are procedures reverse in their actions to the initial operations among the basic set of communication operations. Thus, for instance, the operation of data transmission from a processor to all the available network processors corresponds to the operation of message accumulation by one processor from all the rest processors. As a result, the consideration of the communication procedures should be done pairwise. It is useful as the execution algorithm of the direct and the reverse operations may be in many cases obtained proceeding from the general statements.

Such network topologies as a ring, a two-dimensional grid and hypercube will be further used as examples for the consideration of the basic data transmission operations. For the two-dimensional grid it is also assumed that there are data transmission channels between the bordering processors in the rows and columns of the grid (i.e. the network topology is a torus). As previously, the value m will denote the size of the message in bytes, the value p defines the number of processors in the network, the variable n corresponds to the dimension of the hypercube topology.

3.2.1. Data Transmission between Two Network Processors

The complexity of this communication operation may be obtained by means of substitution of the maximum path (the network diameter – see Table 1.1) into the expression for data transmission time in case of various communication methods (see 3.1.2).

Table 3.1. Data transmission time between two processors

Topology	**Data Transmission**	**Packet Communication**
Ring	$t_н + mt_к \lfloor p/2 \rfloor$	$t_н + mt_к + t_c \lfloor p/2 \rfloor$
Grid-torus	$t_н + 2mt_к \lfloor \sqrt{p}/2 \rfloor$	$t_н + mt_к + 2t_c \lfloor \sqrt{p}/2 \rfloor$
Hypercube	$t_н + mt_к \log_2 p$	$t_н + mt_к + t_c \log_2 p$

3.2.2. Data Transmission from One Processor to All the Network Processors

One-to-all broadcast or *single-node broadcast* (of the same message) is one of the most widely used communication operations. *Single-node accumulation* consists in receiving the messages by one of the processors from all the other network processors. Such operations are used in particular in matrix-vector multiplication implementation, in solving linear equation systems by the Gauss method, as well as in searching for the shortest paths etc.

The simplest way to realize broadcast operation is to carry it out as a sequence of pairwise network processor interactions. However, in case of this approach the greater part of broadcasting is excessive. So a more efficient communication algorithm may be used. First we will describe the message transmission method, and then we will consider packet communication (see 3.1.2).

Message transmission. In case of the **ring** topology the sending processor may initiate data transmission to two neighbors at once. These processors in their turn send the message further in the ring. The complexity of the operation execution in this case will be defined by the following relation:

$$t_{comm} = (t_s + mt_b)\lceil p/2 \rceil.$$

For the **grid-torus** topology the broadcasting algorithm may be obtained basing on the data transmission method, which was used for the ring topology. Thus, broadcasting may be carried out as a two-stage procedure. At the first stage we arrange data broadcasting to all the processors of the network, which are located on the same horizontal line of the grid as the sending processor. During the second stage the processors, which have received the data copy at the first stage, send the messages along the corresponding vertical lines. The estimation of broadcasting duration in accordance to the described algorithm, is defined by the following formula:

$$t_{comm} = 2(t_s + mt_b)\lceil \sqrt{p}/2 \rceil$$

For the **hypercube** broadcasting may be carried out as an *N*-stage data transmission procedure. During the first stage the sending processor sends data to one of the neighbors (for instance, along the first dimension). As a result, there are two processors, which have the copies of the data after the first stage (these results may be also interpreted as bisecting of the initial hypercube and obtaining two identical in size hypercubes of *N-1* dimensionality, each of them has a copy of the initial message). At the second stage the two processors engaged at the first stage send messages to their neighbors along the second dimension etc. As a result of this broadcasting the operation execution time is estimated by the following expression:

$$t_{comm} = (t_s + mt_b)\log_2 p\ .$$

The comparison of the obtained expressions for broadcasting execution duration demonstrates that the hypercube topology shows the best results. Moreover, it is possible to demonstrate that the given result is optimum for the selected communication method based on message transmission.

Packet communication. The broadcast algorithm for the **ring** topology may be obtained by means of logical presentation of the ring structure as a hypercube. As a result, the sending processor sends the data to the processor, which is at *p/2* distance from the initial processor during the first broadcast stage. Further, during the second stage the two processors, which already have the data after the first stage, transmit the data to the processors, which are located at *p/4* distance etc. The time complexity of the broadcast in case of this method is defined by the following expression:

$$t_{comm} = \sum_{i=1}^{\log_2 p}(t_s + mt_b + t_h p/2^i) = (t_s + mt_b)\log_2 p + t_h(p-1)$$

(as previously, if messages are big enough, control data transmission time may be neglected).

For the **grid-torus** topology the broadcast algorithm may be obtained using the method of data transmission applied to the ring network structure. The same way of generalization, that is used for message transmission method, may be also applied. The algorithm, which is obtained, is characterized by the following statement for estimating execution time:

$$t_{comm} = (t_s + mt_b)\log_2 p + 2t_h(\sqrt{p} - 1).$$

The packet communication broadcast algorithm for the **hypercube** (and correspondingly, execution time estimations) does not differ from the variant for message transmission method.

3.2.3. Multinode Broadcast

All-to-all broadcast or *multinode broadcast* is a generalization of a single broadcast operation. *Multinode accumulation* means message reception on every processor from all the network processors. Such operations are broadly used in matrix calculation operations.

A possible way to realize the multinode broadcast operations is to carry out the corresponding set of single-node broadcast operations. However, this approach is not the optimum one for many network topologies, as a part of the necessary single-node broadcast operations may be potentially carried out in parallel. As previously, we will analyze the problem for different data transmission methods (see 3.1.2).

Message transmission. For the **ring** topology each processor may initiate sending its message simultaneously (in any chosen direction in the ring). At any moment of time each processor receives and transmits data. The multinode broadcast operation will be terminated in *(p-1)* data transmission cycles. The duration of the broadcast execution is estimated as:

$$t_{comm} = (t_s + mt_b)(p - 1).$$

For the **grid-torus** topology the multinode data broadcast may be carried out by means of the algorithm, which can be obtained by generalizing the method of data transmission in the ring structure. The scheme of generalization is the following: at the first stage we arrange the message transmission separately to all the network processors located on the same horizontal lines (as a result, enlarged messages of $m\sqrt{p}$ sizes, which unite all the messages on this horizontal line, are formed on every processor of the horizontal line). The stage execution time is

$$t'_{comm} = (t_s + mt_b)(\sqrt{p} - 1).$$

At the second stage the data broadcast is carried out among the processors, which form the vertical lines of the grid. The stage duration is

$$t''_{comm} = (t_s + m\sqrt{p}t_b)(\sqrt{p} - 1).$$

As a result, the total duration of the broadcast operation is defined by the expression:

$$t_{comm} = 2t_s(\sqrt{p} - 1) + mt_b(p - 1).$$

The algorithm of multinode broadcast for the **hypercube** may be obtained by generalizing the previously described method of data transmission for the grid topology to the hypercube of dimensiona *N*. As a result of this generalization the communication scheme appears to be the following: at each stage i, $1 \le i \le N$, of the algorithm execution all the network processors are engaged and exchange data with their neighbors along i dimensionality forming united messages. The broadcast execution time may be obtained by means of the following expression:

$$t_{comm} = \sum_{i=1}^{\log_2 p} (t_s + 2^{i-1} mt_b) = t_s \log_2 p + mt_b(p - 1).$$

Packet communication. The use of the data transmission method, which is efficient for the ring structure and the grid-torus topology, does not improve the execution time of the multinode broadcast. The reason for it is that the generalization of the operation execution algorithms for the single-node broadcast in case of the multinode broadcast leads to overloading of the data transmission channels (i.e. it leads to emerging situations when there are several data packets waiting to be sent at the same moment in the same transmission line). Channel overloading leads to delays in data transmission, and decreases the advantages of the packet transmission method.

The **reduction** problem is a widely spread example of the multinode broadcast. This problem is defined in the most general way, as the procedure of processing data obtained on each processor in the course of the multinode broadcast (as an example of this problem it is possible to consider the problem of computing the sum of values, located on different processors, and transmitting the obtained sum to all the network processors). The ways to solve the reduction problem may be the following:

– the direct approach is to carry out the multinode broadcast operations and then to process the data on each processor separately;

– a more efficient algorithm may be obtained if the single-node data reception operation is used on a separate processor, the data is processed on the processor and the obtained result is sent to all the network processors;

– the optimum way to solve the reduction problem is to combine the multinode broadcast procedures and the data processing operations. Each processor in this case performs the required data processing for the data obtained upon the receiving of the current message (for instance, adds the obtained value to the available on the processor partial sum). The time for solving the reduction problem by means of this algorithm is, in case when the size of the transmitted data has the single length ($m = 1$) and the network topology is a hypercube, defined by the following expression:

$$t_{comm} = (t_s + t_b)\log_2 p \ .$$

The *prefix sum problem*, i.e. the problem of calculation of partial sums S_k of a numerical sequence, is another typical example of the multinode broadcast operation application, where

$$S_k = \sum_{i=1}^{k} x_i, \ \ 1 \le k \le p$$

(we will assume that the number of values coincides with the number of processors, the value x_i is located on i processor, and the result S_k must be obtained on the processor with the number k).

The algorithm for solving the given problem may be also obtained by means of specification of the general method of the multinode broadcast operation execution. In this case the processor performs the summation of the obtained values (but only in the case when the sending processor, which has sent the value, has a smaller number than the receiving processor).

3.2.4. One-to-All Personalized Communication

One-to-all personalized communication or *single-node scatter* means that all the transmitted messages are different. Dual transmission operation for this type of processor interaction is a *single-node gather* of all the messages on one processor from all the network processors (the difference of this operation from the previously described one (single-node accumulation) is that the generalized gather operation does not imply any message interaction (of reduction type) in the process of data transmission).

The complexity of this operation is comparable to the complexity of the multinode data broadcast. The sending processor sends a message of size m to each network processor. Thus, the lower estimation of the operation execution duration may be characterized by the value $mt_b(p-1)$.

Let us analyze the complexity of this broadcast type for the hypercube topology. A possible way to execute the operation is the following: the sending processor transmits half of its messages to one of its neighbors, for instance, along the first dimension. As a result, the initial hypercube is bisected and we obtain two hypercubes of the same size of the dimension *N-1*. Each of them contains half of the initial data. The further operations of message broadcast may be repeated and the total number of repetitions is defined by the initial hypercube dimension. The duration of the operation may be characterized by the following formula:

$$t_{comm} = t_s \log_2 p + mt_b(p-1)$$

(as previously, the complexity of the operation coincides with the duration of the multinode broadcast procedure execution).

3.2.5. Total Data Exchange Operation

Total exchange is the most general case of communication interactions. The need for such operations occurs in Fast Fourier-transform algorithm, matrix transposition etc.

Let us characterize briefly the possible methods of carrying out the total exchange for different data transmission methods (see 3.1.2).

Message transmission. The general scheme of the algorithm for the **ring** topology consists in the following: each processor transmits its initial messages to its neighbor (in any chosen direction in the ring). The processors further receive the transmitted data. Then they choose their messages in the received information. After that they send the remaining part of the data further in the ring. The duration of such data communications is estimated by means of the expression:

$$t_{comm} = (t_s + \frac{1}{2}mpt_b)(p-1) \ .$$

The method of obtaining the data broadcast algorithm for the **grid-torus** topology is the same as the one used for consideration of the other communication operations. At the first stage data transmission is arranged separately for all the processors, which are located on the same horizontal lines (only those initial messages, which should be sent to the processors of the corresponding vertical line, are sent to each processor on the horizontal line). After the termination of the first stage, there are p messages on each processor. These messages should be sent along one of the vertical lines. At the second stage data is sent to the processors, which form the vertical lines. The total duration of these operations is defined by the statement:

$$t_{comm} = (2t_s + mpt_b)(\sqrt{p} - 1) .$$

Total message broadcasting algorithm for the **hypercube** may be obtained by generalizing the method of executing the operation for grid topology to hypercube dimension N. The communication scheme obtained as a result of such generalization is the following: at each stage i, $1 \le i \le N$, of the algorithm execution all the network processorы exchange their data with their neighbors along i-th dimensionality and form united messages. The communication channel in arranging the interaction between two neighbors should be considered as a link between two equal subhypercubes of the initial hypercube. Each processor in the pair sends to the other processor only the messages, which are intended for the neighboring subhypercube processors. The broadcasting time may be estimated by means of the following expression:

$$t_{comm} = (t_s + \frac{1}{2} mpt_b) \log_2 p$$

(except the expenses for broadcasting each processor carries out $mp \log_2 p$ operations of sorting its messages before exchanging the information with its neighbors).

Packet communication. As in case of total broadcasting the application of the data communication method does not lead to the improvement of time characteristics for the total exchange operation. Let us consider, for example, the execution of this communication operation for the hypercube topology network in more detail. In this case broadcasting may be carried out in $p-1$ sequential iterations. All the processors are split into interacting pairs of processors at each iteration. This splitting should be done in such a way that the messages transmitted among the pairs do not use the same transmission paths. As a result, the duration of the total exchange may be determined by the following relation:

$$t_{comm} = (t_s + mt_b)(p-1) + \frac{1}{2} t_h p \log_2 p .$$

3.2.6. Circular Shift

Permutation is a particular case of total exchange. Permutation is the operation of redistributing the information among the network processors, during which each processor transmits a message to another network processor determined in a certain way. The concrete example of permutation is the *circular q-shift*. In this case each processor i, $1 \le i \le N$, transmits the data to the processor $(i+q) \bmod p$. Such a shift operation is used, for instance, in matrix computations.

As circular shift execution for the ring topology may be provided by means of simple data transmission algorithms, we will consider the possible ways of executing this communication operations only for the torus and hypercube topologies with different data transmission methods (see 3.1.2)

Message transmission. The general scheme of the circular shift algorithm for the **grid-torus** topology is described below. Let the processors be enumerated rowwise from 0 to $p-1$. At the first stage the circular shift with the step $q \bmod \sqrt{p}$ at each separate row is executed (if the messages are transmitted through the right borders of the rows during the execution of the shift, then after the completion of this transmission it is necessary to execute one position compensational shift up for the right grid column). At the second stage the circular shift up with the step $\lfloor q / \sqrt{p} \rfloor$ is realized for each grid column. The total duration of all communication operations is determined by the relation:

$$t_{comm} = (t_s + mt_b)(2\lfloor \sqrt{p}/2 \rfloor + 1) .$$

The circular shift algorithm for the hypercube may be obtained by means of logical presentation of hypercube topology as a ring structure. For this we will set the bijective relation between the ring and the hypercube vertices. The necessary relation may be obtained, for instance, be means of the well- known Gray code. This code may be used to determine the hypercube processors, which correspond to particular ring vertices. The mechanism of setting this relation is described in more detail in subsection 3.2. Figure 3.1 shows the hypercube for dimension $N=3$, the corresponding ring vertex is shown for each hypercube processor. The advantage of this relation is the fact that for any two vertices in the ring, the distance between of which is equal to $l=2^i$ for a certain value i, the path between the corresponding hypercube vertices contains only two communication lines (except the case when $i=0$, then the path in the hypercube is of unit length).

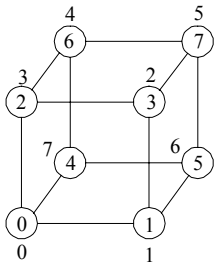

Figure 3.1. The scheme of mapping the hypercube onto the ring (the circles contain the number of processors in the hypercube)

Let us present the value of the shift q as a binary code. The number of non-zero code positions determines the number of stages in the realization scheme of the circular shift operation. The shift is performed at each stage. The value of the shift step is defined by the high-order non-zero position of the value q (for instance, if the initial shift value is $q = 5 = 101_2$, the shift with step 4 is performed at the first stage; at the second stage the step of the shift is equal to 1). The execution of each step (except the shift with step 1) consists in transmission the data along the path, which includes two communication lines. As a result, the upper estimation for the duration of the circular shift execution is determined by the following formula:

$$t_{comm} = (t_s + mt_b)(2\log_2 p - 1).$$

Packet communication. The application of packet communication may increase the efficiency of the circular shift execution for the **hypercube** topology. The realization of all the necessary communication operations may be provided if each processor sends the transmitted data immediately to the receiving processors. The use of the coordinatewise routing (see 3.1.1) helps to avoid collisions when communication lines are used (only one message, which is ready to be sent, can exist for each channel at any given moment of time). The value of the longest path in this case if defined as $\log_2 p - \gamma(q)$, where $\gamma(q)$ is the greatest integer value j, such that 2^j is the divisor of the shift value q. The duration of the circular shift operation may be determined by means of the following expression:

$$t_{comm} = t_s + mt_b + t_h(\log_2 p - \gamma(q))$$

(the time of transmitting the control data may be neglected if the messages are of large sizes and the execution time of the operation may be determined as $t_{comm} = t_s + mt_b$).

3.3. The Method of Logical Presentation of Communication Network

The consideration of the basic communication operations given in subsection 3.1 shows that a number of data transmission algorithms may be described in a simpler way if certain network topologies of interprocessor connections are used. Besides, many communication methods may be obtained by means of a logical presentation of the discussed topology. As a result, the possibility of logical presentation of various topologies on the basis of concrete physical interprocessor structure is essential in parallel computations.

The methods of logical presentation (mapping) of the topologies are characterized by the three following basic characteristics:

- **arc congestion**; it is expressed as the maximum number of arcs of the logical topology mapped onto one communication line of the physical topology;
- **arc dilation;** it is defined as the path of the maximum length in physical topology, onto which the logical topology arc is mapped;
- **vertex expansion;** it is calculated as the relation of the number of vertices in the logical and the physical topologies.

We will consider only the issues of ring and grid mapping onto the hypercube for the topologies discussed in this paper. The approaches for the logical presentation of the topologies described below are characterized by the value 1 for the arc congestion and the arc dilation.

3.3.1. Presentation of the Ring Topology as a Hypercube

Relation between the grid topology and the hypercube may be set with the help of *the binary reflected Gray code* determined in accordance with the following relations:

$$G(0,1) = 0, \quad G(1,1) = 1,$$

$$G(i, s+1) = \begin{cases} G(i,s), & i < 2^s, \\ 2^s + G(2^{s+1} - 1 - i, s), & i \geq 2^s, \end{cases}$$

where i gives the number of the value in the Gray code, and N is the code length. Figure 3.2 illustrates this approach. It shows the mapping of the ring topology onto the hypercube for the network, which consists of $p=8$ processors.

Gray code for N=1	Gray code for N=2	Gray code for N=3	Processor numbers	
			Hypercube	Ring
0	0 0	0 0 0	0	0
1	0 1	0 0 1	1	1
	1 1	0 1 1	3	2
	1 0	0 1 0	2	3
		1 1 0	6	4
		1 1 1	7	5
		1 0 1	5	6
		1 0 0	4	7

Figure 3.2. Mapping of the ring topology onto the hypercube by means of gray code

The essential characteristic of the Gray code is the fact that the neighboring values $G(i,N)$ and $G(i+1,N)$ differ only in a bit position. As a result, the neighboring vertices in the ring topology are mapped onto the neighboring hypercube processors.

3.3.2. Mapping the Grid Topology onto the Hypercube

The mapping the grid topology onto the hypercube may be executed within the frames of the approach, which was used for the ring structure. Then it is necessary to adopt the rule, which says that the grid element with coordinates (i,j) will correspond to the hypercube processor number $G(i,r)||G(j,s)$, where the operation $||$ means Gray code concatenation. This rule may be adopted for mapping the grid $2^r x 2^s$ onto the hypercube of $N=r+s$ dimensionality.

3.4. Estimation of Communication Complexity for Clusters

One of the most efficient methods of organizing the communication network for cluster computer systems (see 1.2.2) is using hubs and switches. In these cases the cluster network topology is the complete graph. There are, however, certain limitations on communication operation simultaneity in the cluster. Thus, data transmission at any given moment of time may be executed only between two processors, if hubs are used. Switches may provide the interactions of several non-intersecting pairs of processors.

Another solution, which is widely used in creating clusters, consists in using packet communication method (which is realized, as a rule, on the basis of TCP/IP protocol). This method is used as the basic means of executing communication operations.

1. If we choose for the further analysis the clusters of this widely used type (the complete graph topology, packet communication method), then the time complexity of the communication operation between two processors may be estimated according to the following formula (*model A*):

$$t_{comm}(m) = t_s + m * t_b + t_h ,$$

the estimation of this type is caused by the expression of packet communication method, when the path length of data transmission is $l = 1$. Such an approach is quite possible. However, it is possible to notice that in this model the time of data preparation t_s is assumed to be constant (it does not depend on the amount of the transmitted data). The time of control data transmission t_h does not depend on the number of the transmitted packets etc. These assumptions do not fully coincide with the real situation, and the time estimations obtained with the help of this model may be not accurate enough.

2. If we take into account all these considerations, we may specify a new enhanced model which provide the estimation of the time transmission complexity for data transmission between two processors in accordance with the following expression (*model B*):

$$t_{comm} = \begin{cases} t_{s_0} + m \cdot t_{s_1} + (m + V_h) \cdot t_b, & n = 1 \\ t_{s_0} + (V_{\max} - V_h) \cdot t_{s_1} + (m + V_h \cdot n) \cdot t_b, & n > 1 \end{cases},$$

where $n = \lceil m/(V_{\max} - V_h) \rceil$ is the number of packets, into which the transmitted message is partitioned, value $V_{\max}$ defines the maximum size of the packet, which may be delivered in the network, and V_h is the volume of control data in each of the transmitted packets. It should be noted that the constant t_{s_0} in the above given relations characterizes the hardware latency component and depends on the parameters of the network equipment being used; the value t_{s_1} defines the time for preparing a data byte for transmission. As a result, the latency value

$$t_s = t_{s_0} + \nu \cdot t_{s_1}$$

increases linearly depending on the amount of the transmitted data. It is assumed that data preparation for transmitting the second and all the further packets may be combined with transmitting the previous packets. Thus, latency cannot exceed the following value

$$t_s = t_{s_0} + (V_{\max} - V_h) \cdot t_{s_1} .$$

Besides latency it is possible to specify the rule for computation of data transmission time in the suggested expressions for estimating the time communication complexity:

$$(m + V_h \cdot n) \cdot t_b .$$

It makes possible to take into account the effect of the increase of the transmitted data amount, if the number of the transmitted packets is increasing due to addition of the control information (packet headers).

3. At the end of the problem analysis it should be noted that it is necessary to estimate the values of the parameters for the relations being used in order to use the above described models in practice. In this respect it may be useful to use simpler methods of computing the time expenses for data transmission. One of the best known schemes of this type is the approach (the Hockney model; see, for instance, Hockney (1994)), which estimates the complexity of the communication between two processors according to the expression (*model C*):

$$t_{comm}(m) = t_s + m t_b .$$

4. In order to check whether these models are adequate to real data transmission processes, we will show the results of the computational experiments carried out in the network of the multiprocessor cluster of Nizhny Novgorod State University (computers IBM PC Pentium 4 1300 Mhz and Fast Ethernet). Communication operations were realized with the help of MPI library in these experiments.

A number of the experiments were carried out for estimating the model parameters:

- the latency value t_s for the models A and C was defined as the time of transmitting the message of zero length;
- the value of bandwidth R was set as the maximal transmission speed observed in the course of experiments, i.e.:

$$R = \max_m \; (t_{comm}(m)/m) ,$$

and it was assumed that $t_b = 1/R$;

- the values t_{s_0} and t_{s_1} were estimated by means of linear approximation of the data transmission time for messages beginning with 0 size up to $V_{\max}$ size,
- there were also set $V_{\max}$ =1500 bytes and V_h =78 bytes.

The data was transmitted between two cluster processors in the experiments. The size of the transmitted messages varied from 0 to 8 Mb. Each operation was carried out repeatedly (more than 100000) in order to obtain more accurate estimations. After that the results were averaged. For illustration the results of an experiment are given below when the size of the transmitted messages varied from 2000 up to 60000 bytes.

The numeric data concerning the errors of the above described models of communication complexity are given in Table 3.2 (the value of the error is given as relative deviation from the real time of data transmission operation execution).

Table 3.2. The errors of the complexity models for data transmission (according to the results of the computational experiments)

Message size (byte)	Transmission time (msec)	The error of the theoretical estimation of data transmission time (%)		
		Model A	Model B	Model C

2000	495	33.45%	7.93%	34.80%
10000	1184	13.91%	1.70%	14.48%
20000	2055	8.44%	0.44%	8.77%
30000	2874	4.53%	-1.87%	4.76%
40000	3758	4.04%	-1.38%	4.22%
50000	4749	5.91%	1.21%	6.05%
60000	5730	6.97%	2.73%	7.09%

The results of the experiments demonstrate that the estimations of data transmission complexity according to model B have the least error.

It should be also noted that model C accuracy may appear to be sufficient for the preliminary analysis of the time expenses on communication operations. Besides, this model is the simplest one among all the models, which have been discussed here. With regard to the latter fact we will be using model C (the Hockney model) in all the further sections for estimating the data transmission complexity. Moreover, further we will use the following format for the Hockney model (see Hockney (1994)):

$t_{comm}(m) = \alpha + m/\beta$,

where α is the latency of the data transmission network (i.e. $\alpha = t_s$), β is the network bandwidth (i.e. $\beta = R = 1/t_b$).

3.5. Summary

This section is devoted to estimation of communication complexity for parallel algorithms.

Subsection 3.1 gives the general characteristics of the routing algorithms and data transmission methods. The method of *store-and-forward routing* and *cut-through routing* method are analyzed in detail. The time of communication operations is estimated for these methods.

Subsection 3.2 defines the basic types of data transmission operations, which are carried out in the course of parallel computations. The following communication operations are referred to as the basic ones:

- *data transmission among the network processors*,
- *one-to-all broadcast* or *single-node broadcast* and the reverse operation, i.e. *single-node accumulation*,
- *all-to-all broadcast* or *multinode broadcast* and the reverse operation, i.e *multinode accumulation*,
- *one-to-all personalized communication*[1)] or *single-node scatter* and the reverse operation, i.e *single-node gather*,
- *total exchange*, i.e. generalized data transmission from all the procerros to all the processors of the network.

Data transmission for all the mentioned above operations are considered in terms of the ring, the grid and the hypercube topologies. For each of the algorithms described there are estimations of the complexity time both for data transmission and packet communication.

Subsection 3.3 describes the methods of logical presentations of topologies on the basis of physical interprocessor structures. The use of logical topologies simplifies the description of a number of data transmission algorithms, decreases expenses on communication operations etc.

Subsection 3.4 describes the models, which may help to estimate the time of data transmission operations for cluster computer systems. For comparing the accuracy of different time estimations a number of computational experiments have been carried out. The results of the experiments are described in the subsection. The results of the experiments makes possible to find the most precise model (model B). Besides, it is noted that for the preliminary analysis of the time communication complexity a simpler model (the Hockney model) may be more suitable.

3.6. References

The works by Kumar (1994) and Quinn (2004) may be recommended as additional teaching material on the problems discussed in the subsection.

Creating models for estimating the time of communication operations is widely discussed in many papers. Such works as Culler, et al. (1996), Skillicorn and Talia (1998), Andrews (2000) might be of use. The Hockney model was first published in Hockney (1994). Model B from subsection 3.4 is presented in Gergel, Strongin (2001).

[1)] The generalization of operation consists in broadcasting various messages to different processors; for the reverse reception operation all the accumulated messages are also different.

3.7. Discussions

1. What basic characteristics are used for the estimation of the data transmission network topology? Give the values of the characteristics for the following types of communication structures (a complete graph, a linear array, a grid etc.)
2. What basic methods are applied to routing the data transmitted in the network?
3. What are the basic methods of data transmission? Give the analytical estimations of the execution time for these methods.
4. What data transmission operations may be selected as the basic ones?
5. What are the execution algorithms of one-to-all broadcast for the ring, the grid and the hypercube topologies? Give the estimations of the time complexity for these algorithms.
6. What are the execution algorithms of all-to-all broadcast for the ring, the grid and the hypercube topologies? Give the estimations of the time complexity for these algorithms.
7. What are the possible execution algorithms of reduction? Which of them is the best as far as the execution time is concerned?
8. What does the execution algorithm of the circular shift consist in?
9. Why is it efficient to use logical topologies? Give the examples of the algorithms for logical presentation of communication network structure.
10. How do the models for estimating the execution time of data transmission in cluster computer systems differ from one another? Which model is the most accurate? Which of them may be used for the preliminary analysis of the time comlexity of the communication operations?

3.8. Tasks and Exercises

1. Develop the execution algorithms of the basic data transmission operations for the network topology in the form of a three-dimensional grid.
2. Develop the execution algorithm of the basic data transmission operations for the network topology in the form of a binary tree.
3. Use model B from subsection 3.4 for the estimation of the time complexity of data transmission operations. Compare the obtained results.
4. Use model C from subsection 3.4 for the estimation of the time complexity of data transmission operations. Compare the obtained results.
5. Develop the algorithms of logical presentation of the binary tree for various physical network topologies.

References

Gergel, V.P., Strongin, R.G. (2001, 2003 - 2 edn.). Introduction to Parallel Computations. - N.Novgorod: University of Nizhni Novgorod (In Russian)

Culler, D.E., et al. (1996). LogP: A practical model for parallel computation. – Comm. Of the ACM, 39, 11, pp. 75-85.

Hockney, R. (1994). The communication challenge for MPP: Intel Paragon and Meiko CS-2. – Parallel Computing, 20 (3), pp. 389-398.

Kumar V., Grama, A., Gupta, A., Karypis, G. (1994). Introduction to Parallel Computing. - The Benjamin/Cummings Publishing Company, Inc. (2nd edn., 2003)

Quinn, M. J. (2004). Parallel Programming in C with MPI and OpenMP. – New York, NY: McGraw-Hill.

Skillicorn, D.B., Talia, D. (1998). Models and languages for parallel computation. – ACM Computing surveys, 30, 2.