A-41

A-41

Improved Multimessage Multicasting Approximation Algorithms

T. F. Gonzalez
Department of Computer Science
University of California
Santa Barbara, CA, 93106

Abstract

We consider Multimessage Multicasting over the n processor complete (or fully connected) static network (MM_C). We present a fast approximation algorithm with an improved approximation bound for problem instances with small fan-out (maximum number of processors receiving any given message), but arbitrary degree d, where d is the maximum number of messages that each processor may send (receive). These problem instances are the ones that arise in practice, since the fan-out restriction is imposed by the applications and the number of processors available in commercial systems.

Our algorithms are centralized and require all the communication information ahead of time. Applications where this information is available include iterative algorithms for solving linear equations and most dynamic programming procedures. The Meiko CS-2 machine and in general computer systems with processors communicating via dynamic permutation networks whose basic switches can act as data replicators (e.g., n by n Benes network with 2 by 2 switches that can also act as data replicators) will also benefit from our results at the expense of doubling the number of communication phases.

1 Introduction

1.1 The MM_C Problem

The Multimessage Multicasting problem over the n processor static network (MM_C) consists of finding a communication schedule with least total communication time for multicasting (transmitting) any given set of messages. Specifically, there are n processors, $P = \{P_1, P_2, \ldots, P_n\}$, interconnected via a network N. Each processor is executing processes, and these processes are exchanging messages that must be routed through the links of N. Our objective is to determine when each of these messages is to be transmitted so that all the communications can be carried in the least total amount of time. Our introduction is a condensed version of the one in [7], which includes a complete justification for the MM_C problem as well as motivations, applications and examples.

Routing in the complete static network (there are bidirectional links between every pair of processors) is the simplest and most flexible when compared to other static and dynamic networks. Multimessage Multicasting for dynamic networks that can realize all permutations and replicate data (e.g., n by n Benes network based on 2 by 2 switches that can also act as data replicators) is not too different, in the sense that the number of communication phases for these dynamic networks can be shown to be twice of that in the complete network. This is accomplished by translating each communication phase for the complete network into two communication phases for these dynamic networks. The first phase replicates data and transmits it to other processors, and the second phase distributes data to the appropriate processors ([13], [14], [16]). The IBM GF11 machine [1], and the Meiko CS-2 machine use Benes networks for processor interconnection. The two stage translation process can also be used in the Meiko CS-2 computer system, and any multimessage multicasting schedule can be realized by using basic synchronization primitives. This two step translation process can be reduced to one step by increasing the number of network switches about 50% ([13], [14], and [16]). In what follows we concentrate on the MM_C problem because it has a simple structure, and, as we mentioned before, results for this network can be easily translated to other dynamic networks.

Let us formally define our problem. Processor P_i needs to multicast s_i messages, each requiring one time unit to reach any of its destinations. The j^{th} message of processor P_i has to be sent to the set of processors $T_{i,j} \subseteq P - \{P_i\}$. Let r_i be the number of distinct messages that processor P_i should receive. We define the *degree* of a problem instance as $d = \max\{s_i, r_i\}$, i.e., the maximum number of messages that any processor sends or receives. We define the *fan-out* of a problem instance as $k = \max\{ \mid T_{i,j} \mid \}$, i.e., the maximum number of different processors that must receive any given message. Consider the following example.

Example 1.1 *There are three processors ($n = 3$). Processors P_1, P_2, and P_3 must transmit 3, 4 and 2 messages, respectively (i.e., $s_1 = 3, s_2 = 4$, and $s_3 = 2$). The destinations of these messages is: $T_{1,1} = \{2\}$, $T_{1,2} = \{3\}$, $T_{1,3} = \{2,3\}$, $T_{2,1} = \{1\}$, $T_{2,2} = \{1\}$, $T_{2,3} = \{3\}$, $T_{2,4} = \{1,3\}$, $T_{3,1} = \{1,2\}$, and $T_{3,2} = \{2\}$. For this example $r_1 = 4$, $r_2 = 4$, and $r_3 = 4$.*

It is convenient to represent problem instances by directed multigraphs. Each processor P_i is represented by the vertex labeled i, and there is a directed edge (or branch) from vertex i to vertex j for each message that processor P_i needs to transmit to processor P_j. The $|T_{i,j}|$ directed edges or branches associated with each message are *bundled* together, i.e., belong to a set for the bundle.

The communications allowed in our complete network satisfy the following two restrictions.

1.- During each time unit each processor may transmit one message, but such message can be multicasted to a set of processors; and

2.- During each time unit each processor may receive at most one message.

Our communication model allows us to transmit any of the messages in one or more stages. I.e., each set $T_{i,j}$ can be partitioned into subsets, and each of these subsets is transmitted at a different time. This added routing flexibility reduces the total communication time.

A *communication mode* C is a collection of subsets of branches from a subset of the bundles that obey the following communications rules imposed by our network:

1.- Branches may emanate from at most one of the bundles in each processor; and

2.- All of the branches end at different processors.

A *communication schedule* S for a problem instance I is a sequence of communication modes such that each branch in each message is in exactly one of the communication modes. The *total communication time* is the latest time at which there is a communication which is equal to the number of communication modes in schedule S, and our problem consists of constructing a communication schedule with least total communication time. From the communication rules we know that a degree d problem instance has at least one processor that requires d time units to send, and/or receive all its messages. Therefore, d is a trivial lower bound for the total communication time. To simplify the analysis of our approximation bound we use this simple measure. Another reason is that load balancing (placement) and multimessage multicasting (routing) are normally separate procedures, and load balancing must use a simple objective function in terms of the problem instance it generates that somehow represents the final communication time for the particular placement and some reasonable routing procedure.

Using our multigraph representation one can visualize the MM_C problem as a generalized edge coloring directed multigraph (GECG) problem. This problem consists of coloring the edges with the least number of colors (positive integers) so that the communication rules (now restated in the appropriate format) imposed by our network are satisfied: (1) every pair of edges from different bundles emanating from the same vertex must be colored differently; and (2) all incoming edges to each vertex must be colored differently. The colors correspond to different time periods. In what follows we corrupt our notation by using interchangeably colors and time periods; vertices and processors; and bundles, branches or edges, and messages.

1.2 Previous Work, New Results, and Applications

Gonzalez [7] developed an efficient algorithm to construct for any degree d problem instance a communication schedule with total communication time at most d^2, and presented problem instances for which this upper bound on the communication time is best possible, i.e. the upper bound is also a lower bound. One observes that the lower bound applies when the fan-out is huge, and thus the number of processors is also huge. Since this environment is not likely to arise in the near future, we turn our attention in subsequent sections to important subproblems likely to arise in practice.

The *basic multicasting problem* (BM_C) consists of all the degree $d = 1$ MM_C problem instances, and can be trivially solved by sending all the messages at time zero. There will be no conflicts because $d = 1$, i.e., each processor must send at most one message and receive at most one message. When the processors are connected via a dynamic network whose basic switches allow data replication, the basic multicasting problem can be solved in two stages: the data replication step followed by the data distribution step ([13], [16], [14]).

This two stage process can be used in the MEIKO CS-2 machine [7].

Gonzalez [7] also considered the case when each message has fixed fan-out k. When $k = 1$ (multimessage unicasting problem MU_C), Gonzalez showed that the problem corresponds to the Makespan Openshop Preemptive Scheduling problem which can be solved in polynomial time, and each degree d problem instance has a d color optimal coloration. The interesting point is that each communication mode translates into a single communication step for processors interconnected via permutation networks (e.g., Benes Network, Meiko CS-2, etc.), because in these networks all possible one-to-one communications can be performed in one communication step.

It is not surprising that several authors have studied the MU_C problem as well as several interesting variations for which NP-completeness has been established, subproblems have been shown to be polynomially solvable, and approximation algorithms and heuristics have been developed. Coffman, Garey, Johnson and LaPaugh [2] studied a version the multimessage unicasting problem when messages have different lengths, each processor can send (receive) $\alpha(P_i) \geq 1$ ($\beta(P_i) \geq 1$) messages simultaneously, and messages are transmitted without interruption (non-preemptive mode). Whitehead [18] considered the case when messages can be sent indirectly. The preemptive version of these problems as well as other generalizations were studied by Choi and Hakimi ([4], [5], [3]), Hajek and Sasaki [11], Gopal, Bongiovanni, Bonuccelli, Tang, and Wong [10]. Some of these papers considered the case when the input and output units are interchangeable, i.e., each processor can be involved in at most $\gamma(P_i)$ message transmissions (sending and/or receiving). Rivera-Vega, Varadarajan and Navathe [15] studied, the file transferring problem, a version the multimessage unicasting problem for the complete network when every vertex can send (receive) as many messages as the number of outgoing (incoming) links. Our MM_C problem is closest to the communication model in the Meiko CS2 machine and it involves multicasting rather than just unicasting.

The MM_C problem is significantly harder than the MU_C. Gonzalez [7] showed that even when $k = 2$ the decision version of the MM_C problem is NP-complete. He also developed an algorithm to construct a communication schedule with total communication time $2d-1$ for the case when the fan-out is two, i.e., $k = 2$. Gonzalez [7] developed an $O(q \cdot d \cdot e)$ time algorithm, where $e \leq nkd$ (the input size), to construct for problem instances of degree d a communication schedule with total communication time $qd + k^{\frac{1}{q}}(d-1)$, where q is the maximum number of colors that can be used to color each bundle and $k > q >= 2$.

We present a fast approximation algorithm with an improved approximation bound for problems instances with any arbitrary degree d, but small fan-out (maximum number of processors that may receive a given message), where d is the maximum number of messages that each processor may send (receive). These problem instances are the ones that arise in practice, since the fan-out restriction is imposed by the applications and the number of processors available in commercial systems.

The MM_C problem arises when solving sparse systems of linear equations via iterative methods (e.g., a Jacobi-like procedure), and most dynamic programming procedures.

2 Improved Approximation Algorithm

All of our approximation algorithms generate a coloration with at most $a_1 \cdot d + a_2$ colors. The value of constant a_1 for the different methods we have developed and for different values for k is given in Table 1. The methods labeled "simple" are for the method described in [7]. The "involved (2c)" is the method discussed in this paper. The "2c" stands for at most two colors per bundle. We briefly discuss in Section 4 and in [8] the remaining methods.

Table 1: Number of Colors For The Different Methods.

Method \ k	3	5	10	20	100
Simple (2c)	3.73	4.23	5.16	6.47	12.00
Involved (2c)	3.33	3.60	4.60	6.00	11.54
Matching (2c)	2.67	3.50	4.50	6.00	11.53
Better Bound	2.50	3.50	4.40	5.75	11.52
Simple (3c)	–	4.00	4.81	5.60	7.62
Involved (3c)	–	4.00	4.67	5.20	7.24
Simple (4c)	–	5.50	5.78	6.11	7.16
Simple (5c)	–	–	6.58	6.82	7.51

The input to our algorithm is a directed multigraph G with bundled edges, integers h and l that restrict the color selection process and it is assumed that ($k > l > h \geq 1$). Note that k and d can be extracted from the multigraph. The algorithm colors the edges emanating out of P_1, then P_2, and so on until P_{j-1}. Then the algorithm will color the edges emanating out of P_j. Each of these branches leads to a processor with at most $d-1$ other edges incident to it, some of which have already been colored. These colors are called t_{j-1}*-forbidden* with respect to a given branch emanating from P_j. When considering processor P_j, a t_{j-1}*-forbidden* color with a special property is selected from each bundle and then such color is used to color as many of the branches of the bundle. The remaining uncolored branches are colored with a second color whose existence is guaranteed by setting the total number of colors available to an appropriate number. Before we present our algorithm we define some useful terms.

At the beginning of the j^{th} iteration the algorithm has colored all the branches emanating from processors $P_1, P_2, \ldots, P_{j-1}$. Let us define the following terms from this partial coloration. For $0 \leq i \leq k$, let C_i^b be the set of colors that are t_{j-1}-forbidden in exactly i branches of bundle b. Let $c_i^b = |C_i^b|$. When the set b is understood, we will use c_i for c_i^b, and C_i for C_i^b. Since there can be at most $d-1$ t_{j-1}-forbidden colors in each branch and there are at most k branches in each bundle, it then follows that $\sum_{i=1}^{k} iC_i^b \leq (d-1)k$ for each bundle b emanating from P_j. Clearly, all the branches of bundle b can be colored with any of the colors in C_0^b that have not been used to color other branches emanating from P_j. Also, one can color all the branches of bundle b with two colors, $a \in C_i^b$ and $b \in C_j^b$ provided that colors a and b are not t_{j-1}-forbidden in the same branch of bundle b, and have not been used to color another branch emanating from processor P_j. Just after coloring a subset of branches of a bundle emanating from processor P_j, we say that a color is s_j*-free* if such color has not yet been used to color any of the branches emanating from processor P_j.

To simplify our notation we define the expressions L and R as follows

$$L = \frac{h^2+h+2}{2} + \frac{l}{d-1} - \frac{h^2+h-2}{2(d-1)}, \text{ and}$$
$$R = (h+1)^2 + \frac{(h+1)(h^2+3h)}{2(l-h)} + \frac{-2lh^2+h^3+h}{2(d-1)(l-h)}$$

Procedure Coloring is defined for all $d \geq \frac{2l+2h^2}{h^2+3h-2}$, $k \geq L$, $k > l > h \geq 1$ and $d > 4$. These preconditions might give the feeling that there are a large number of cases for which our algorithm is not defined, but this is not a the case because for each $k \geq 3$ there is a nonempty set of h and l values for which it is defined. We begin by establishing in Lemma 2.1 that $L \leq R$. This fact will be used to partition in two cases the set of values for which our algorithm is defined. For brevity we do not include proofs for our lemmas. These proofs appear in [8], and also include Mathematica programs (and their outputs) for the mechanical parts of these proofs.

Lemma 2.1 *For the set of values Procedure Coloring is defined* $L \leq R$.

In Table 3 we define equations eq.(0), ..., eq.($h+1$) that are used by the algorithm and are necessary for the correctness proof (Theorem 2.1).

Table 3: Equations eq.(0), ..., eq.($h+1$).

$$c_0 \geq d; \qquad eq.(0)$$

$$\text{For } 1 \leq j \leq h \quad \sum_{i=0}^{j} c_i \geq (j+2)d - 2j; \qquad eq.(j)$$

$$\sum_{i=0}^{l} c_i \geq (h+2)d - 2h. \qquad eq.(h+1)$$

Let us now briefly outline our Procedure as well as some of the arguments used in the correctness proof. When coloring the bundles emanating out of processor P_j, Procedure Coloring finds the smallest integer q_b such that equation eq.(q_b) holds. Lemma 2.5 shows that at least one of such equation holds for each bundle b. Then r_b is defined as $min\{q_b, h\}$. For each bundle b emanating from processor P_j an s-free color from $C_0^b, C_1^b, \ldots, C_{r_b}^b$ is selected to color as many branches in bundle b as possible. Lemma 2.4 can be used to show that one such colors exist. The integer s_b is defined as q_b if $0 \leq q_b \leq h$ and it is set to l otherwise. The remaining uncolored branches of each bundle b are colored with an s-free color in $C_0^b, C_1^b, \ldots, C_{s_b}^b$. One can show that the existence of such color from Lemma 2.5.

Procedure Coloring is given below. For the set of valid inputs defined above, procedure computes the maximum number of colors needed (Δ) and a coloration for G with at most Δ colors.

Procedure Coloring (G, h, l)
/* k, d, L, and R can be computed from G */
/* Procedure is defined for $d \geq \frac{2l+2h^2}{h(h+3)}$,
$k \geq L$, $k > l > h \geq 1$, and $d \geq 4$ */
$\Delta = \frac{((2d-4)h+4d-2)l+2(d-1)k+(2-d)h^2+(d-2)h+2d}{2(l+1)}$;
if $R \leq k$ **then** $\Delta = \frac{d(k+h+1)-(k+h)}{h+1}$;
// Theorem 2.1 establishes correctness.//
for each processor P_j **do**
for each bundle b emanating from P_j **do**
compute $C_0^b, C_1^b, C_2^b, \ldots C_k^b$;
let q_b be the smallest integer such that
equation eq.(q_b) holds;
let $r_b =$ **min** $\{q_b, h\}$;

let $s_b = q_b$ if $0 \le q_b \le h$ and $s_b = l$ otherwise;
endfor
/* Color a subset of edges emanating from each bundle of P_j. **for-loop-a** */
for each uncolored bundle b of P_j **do**
color as many branches of bundle b with one s_j-free color in $C_0^b, C_1^b, \ldots, C_{r_b}^b$;
/* Color the remaining uncolored edges emanating from P_j. **for-loop-b***/
for each partially colored bundle b of P_j
color all uncolored branches of b with an s_j-free color in $C_0^b, C_1^b, \ldots, C_{s_b}^b$;
endfor;
end of Procedure Coloring

To establish that Procedure Coloration generates a valid coloration for the cases it is defined is difficult. Theorem 2.1 establishes that our algorithm generates valid colorations for all valid inputs, and that it takes $O(ed)$, where e is the total number of edges. The proof of this theorem is based on Lemmas 2.4 and 2.5. Lemma 2.5 established that at least one of the equations $eq.(j)$ holds for each bundle emanating out of processor P_j, and Lemma 2.4 is used to show that one color from each bundle can be selected to color a subset of its branches. These lemmas are then used to show that a second color exists to color the remaining branches of each partially colored bundle. Lemma 2.3 is used in the proof of Lemmas 2.4 and 2.5, and requires Lemma 2.2.

Lemma 2.2 *For the set of values Procedure Coloring is defined* $R \ge (h+1)^2 - \frac{h^2}{d-1} + \frac{h+1}{d-1}$.

Lemma 2.3 *The value for Δ defined by Procedure Coloration is greater than or equal to* $(h+2)d - 2h$.

Lemma 2.4 *At the beginning of the j^{th} iteration of Procedure Coloring each bundle b emanating from processor P_j satisfies* $\sum_{i=0}^{h} c_i^b \ge d$.

Lemma 2.5 *At the beginning of the j^{th} iteration of Procedure Coloring each bundle b emanating from processor P_j satisfies at least one of the inequalities $eq.(j)$, for $0 \le j \le h+1$, holds.*

Theorem 2.1 *Procedure Coloring generates a communication schedule with total communication equal to the value of Δ computed by the algorithm, for every instance for which the Procedure is defined. The time complexity of the procedure is $O(ed)$, where e is the total number of edges.*

Proof: First we prove that Procedure Coloration colors all the edges in the multigraph with Δ colors, where Δ is determined by the algorithm. Then we establish the time complexity bound.

Consider now the iteration for P_j for any $1 \le j \le n$. By Lemma 2.5 we know that at least one of the equations eq.(i) for $0 \le i \le h+1$ holds for each bundle emanating from P_j. Therefore, all the q_b values are integers in the range $[0,h+1]$, and all the r_b values are integers in the range $[0,h]$.

We now claim that one can color a nonempty subset of branches from each bundle with a distinct s-free color in $C_0^b, C_1^b, \ldots C_{q_j}^b$. We prove this by showing that $\sum_{i=0}^{r_b} c_i^b \ge d$, since this fact guarantees that one unique s-free color in $C_0^b, C_1^b, \ldots C_{q_j}^b$ for each bundle b can be selected in for-loop-a to color a nonempty subset of edges emanating out of each bundle. As we established before, $r_b \le h$. If $r_b = h$ then by Lemma 2.4 it follows that $\sum_{i=0}^{r_b} c_i^b \ge d$. On the other hand, if $r_b < h$ then by definition of r_b and Lemma 2.5 we know that $eq.(r_b)$ holds. This implies that either $c_0 \ge d$ or $\sum_{i=0}^{r_b} c_i \ge (r_b+2)d - 2r_b$. Since $d > 2$, it then follows that $\sum_{i=0}^{r_b} c_i^b \ge d$. Therefore, in for-loop-a one can select unique s-free color in $C_0^b, C_1^b, \ldots C_{q_j}^b$ for each bundle b to color a nonempty subset of edges emanating out of each bundle.

We now claim that at each iteration in the for-loop-b one can select unique colors to color the remaining uncolored branches of each bundle. From the definition of s_b we know that $\sum_{i=0}^{s_b} c_i \ge (s_b+2)d - 2s_b$. The number of colors that were t_{j-1}-forbidden in the same branch as the color selected in for-loop-b is at most $(d-2) \cdot r_b$, and the maximum number of colors used during for-loop-a and for-loop-b is at most $2d-1$. It follows that the colors that one can use to color the remaining branches are at least $(s_b+2)d - 2s_b - (d-2) \cdot r_b - 2d + 1$. This is equivalent to $(d-2)(s_b - r_b) + 1$. Since $d > 2$ and $s_b \ge r_b$, we know that there is at least one color left with which we can color all the remaining uncolored branches. This completes the correctness proof.

It is simple to see that the time complexity of the algorithm is bounded by $O(ed)$, where e is the total number of edges in the graph. This follows from the observation that each edge is considered a constant number of times and each time the algorithm spends $O(d)$ time on it.

□

Summary

A more involved analysis can establish a slightly smaller approximation bound (see Table 1 "Better Bound"), but it is asymptotic to the ones in this paper. The proof for this bound is tedious. By selecting three colors instead of two colors, one can also improve the approximation bounds in this paper. The time complexity is the same as the corresponding one in the previous two procedures, but the approximation bound is better. For brevity we cannot provide the details of these procedures, so we just point out that their benefit is when k is about 15 (about 10% improvement) and there is a benefit of more than 50% when k is about 100. The proofs are similar in nature to the ones in the previous sections, however the equations are much more complex.

Acknowledgements

We would like to thank Professor Si-Qing Zheng from LSU for presenting our paper at the PDCS'96 Conference.

References

[1] G. S. Almasi, and A. Gottlieb, *Highly Parallel Computing*, The Benjamin/Cummings Publishing Co., Inc., New York, 1994.

[2] E. G. Coffman, Jr, M. R. Garey, D. S. Johnson, and A. S. LaPaugh, Scheduling File Transfers in Distributed Networks, *SIAM J. on Computing*, 14(3) (1985), pp. 744 - 780.

[3] H.-A. Choi, and S. L. Hakimi, Data Transfers in Networks, *Algorithmica*, Vol. 3, (1988), 223 - 245.

[4] H.-A. Choi, and S. L. Hakimi, Scheduling File Transfers for Trees and Odd Cycles, *SIAM J. on Comp.*, Vol. 16, No. 1, February 1987, pp. 162 - 168.

[5] H.-A. Choi, and S. L. Hakimi, "Data Transfers in Networks with Transceivers," *Networks*, Vol. 17, (1987), pp. 393 - 421.

[6] Gonzalez, T. F., "Multi-Message Multicasting," Proceedings of the Third International Workshop on Parallel Algorithms for Irregularly Structured Problems (Irregular'96), to appear.

[7] Gonzalez, T. F., "Multimessage Multicasting: Complexity and Approximations," UCSB TRCS-96-15, July 1996.

[8] Gonzalez, T. F., "Proofs for Improved Approximation Algorithms for Multimessage Multicasting," UCSB TRCS-96-17, July 1996, (or http://www.cs.ucsb.edu/~teo).

[9] T. F. Gonzalez, and S. Sahni, Open Shop Scheduling to Minimize Finish Time, *JACM*, Vol. 23, No. 4, October 1976, pp. 665 - 679.

[10] I. S. Gopal, G. Bongiovanni, M. A. Bonuccelli, D. T. Tang, and C. K. Wong, An Optimal Switching Algorithm for Multibean Satellite Systems with Variable Bandwidth Beams, *IEEE Transactions on Comm.*, COM-30, 11 (1982) pp. 2475 - 2481.

[11] B. Hajek, and G. Sasaki, Link Scheduling in Polynomial Time, *IEEE Transactions on Information Theory*, Vol. 34, No. 5, Sept. 1988, pp. 910 - 917.

[12] I. Holyer, The NP-completeness of Edge-Coloring, *SIAM J. Comp.*, 11 (1982), 117 - 129.

[13] T. T. Lee, Non-blocking Copy Networks for Multicast Packet Switching, *IEEE J. Selected Areas of Comm.*, Vol. 6, No 9, Dec. 1988, pp. 1455 - 1467.

[14] S. C. Liew, A General Packet Replication Scheme for Multicasting in Interconnection Networks, *Proceedings IEEE INFOCOM '95*, Vol.1 (1995), pp. 394 - 401.

[15] P. I. Rivera-Vega, R, Varadarajan, and S. B. Navathe, "Scheduling File Transfers in Fully Connected Networks," *Networks*, Vol. 22, (1992), pp. 563 - 588.

[16] J. S. Turner, A Practical Version of Lee's Multicast Switch Architecture, *IEEE Transactions on Communications*, Vol. 41, No 8, Aug. 1993, pp. 1166 - 1169.

[17] V. G. Vizing, On an Estimate of the Chromatic Class of a p-graph, *Diskret. Analiz.*, 3 (1964), pp. 25 - 30 (In Russian).

[18] J. Whitehead, The Complexity of File Transfer Scheduling with Forwarding, *SIAM J. on Computing* Vol. 19, No 2, April 1990, pp. 222 - 245.