PERGAMON

Computers & Operations Research 28 (2001) 1149–1164

computers & operations research

www.elsevier.com/locate/dsw

A note on distributed multicast routing in point-to-point networks

Roman Novak*, Jože Rugelj, Gorazd Kandus

Department of Digital Communications and Networks, Jozef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia

Received 1 December 1998; received in revised form 1 September 1999

Abstract

The distributed algorithm for a multicast connection set-up, based on the 'cheapest insertion' heuristic, is reviewed. The multicast routing problem is translated into a Steiner tree problem in point-to-point networks where nodes have only a limited knowledge about the network. A solution is proposed in which the time complexity and the amount of information exchanged between network nodes are proportional to the number of members of the multicast group. The Steiner tree is constructed by means of a distributed table-passing algorithm. The analysis of the algorithm presented, backed up by simulation results, confirms its superiority over the algorithm based on 'waving technique'.

Scope and purpose

Multicasting is a mechanism used in communication networks that allows distribution of information from a single source to multiple destinations. The problem of finding a multicast connection for a static group of communicating entities in connection-oriented point-to-point network can be formulated in graph theory as a minimum Steiner tree problem. Due to NP-completeness of the Steiner tree problem multicast, routing algorithms are based on heuristics. The diversity of network environments and the lack of centralised information about network topology require an effective distribution of the multicast routing algorithms among the network nodes. This article presents an alternative to the distributed algorithm proposed by Rugelj and Klavzar that implements the same heuristics for the construction of a minimum cost multicast connection in point-to-point networks. The present algorithm constitutes a substantial improvement over that previously proposed with regard to running time and the amount of the information exchanged between network nodes. © 2001 Elsevier Science Ltd. All rights reserved.

Keywords: Steiner tree problem; Multicast communication; Distributed algorithm

* Corresponding author. Tel.: + 386-1477-3724; fax: + 386-1426-2102.
E-mail address: roman.novak@ijs.si (R. Novak).

1. Introduction

Techniques of transmitting messages to multiple users in a multicast communication session have evolved rapidly over the past few years. Broadcasting is a technique of transmitting a packet from a source node to every other node in a network. It guarantees that every node in a network receives one and only one copy of a packet. Multicasting, i.e., selective broadcasting, is the transmission of a packet from a source node to a limited set of destination nodes, the multicast group. It provides additional bandwidth savings over broadcasting, for applications where the multicast group consists of a subset of the network nodes.

One of the key issues to be addressed is that of routing multicast communications. In order to be efficient, tree-shaped topologies of multicast connections are required. A minimum number of data packets can be transmitted in parallel to various destinations along the branches of the tree, with duplication carried out only where the tree branches. When the utilisation of network resources is the main consideration, the problem of finding the topology of connections reduces to the well-known Steiner tree problem in graphs [1,2].

The success of a routing algorithm is predicated on having an accurate view of the state of the network, including its topology and the availability of resources at every node. When the nodes do not have complete knowledge of the topology and state of the network, distributed routing algorithms are needed [3]. A distributed route computation would presumably take place on those nodes having the resources, and the resulting decisions would become effective immediately at those nodes. The distributed algorithm is a collection of asynchronous, independent node algorithms, one per node in a network. The distributed algorithm establishes multicast connection in a decentralised manner, with each node carrying out a specified routing algorithm.

In this article we present a distributed algorithm as an alternative to that presented in Rugelj and Klavžar [4]. The proposed algorithm is superior to the latter, and to the algorithms proposed earlier [5,6], as regards average case behaviour concerning the running time and the number of exchanged messages. The layout of the paper is as follows. Section 2 briefly introduces a network model. The importance of efficient utilisation of network resources as well as the frequently considered delay optimisation criteria are also discussed. Section 3 deals with the Steiner tree problem and with its application to communication networks. The heuristic algorithm on which the distributed algorithms are based is presented in Section 4. Section 5 provides a short overview of the distributed algorithm presented in [4]. A new distributed algorithm is presented in Section 6 and the results of performance measurements in Section 7. The appendix contains the formal specification of the proposed algorithm in Z specification language.

2. Network model

Multicast routing algorithms can be partitioned into three classes: shortest path-based algorithms, Steiner-based algorithms, and constrained Steiner-based algorithms [7]. The shortest path-based algorithms are suitable for those applications, such as videoconferences, which require a short propagation delay between the source and each of the receivers [8]. The Steiner-based algorithms are well suited for those services, such as VoD, which tend to consume as few network resources as possible [9–11]. The constrained Steiner-based algorithms combine both types of requirements in a single multicast communication [12,13]. Steiner-based algorithms solve much harder problem (NP-complete) than shortest path-based algorithms. Constrained Steiner-based algorithms are usually derived by introducing additional constraints into Steiner-based algorithms. Therefore, research into efficient Steiner-based algorithms is of high importance.

Steiner-based algorithms are becoming important in the mobile wireless networks of the future, where bandwidth may be limited if not scarce. Further, multicast routing problem in wide area WDM networks is Steiner-based, as the costs of wavelength conversion at nodes and of using wavelengths on links have to be considered.

The revised distributed Steiner-based algorithm presented here is based on a network model that can be represented as an undirected edge-weighted graph. The network model is taken from Rugelj and Klavžar [4] for the sake of comparability and simplicity, although the proposed algorithm is not limited to undirected networks. The algorithm can be used without modification for building source-based trees instead of shared trees. The cost efficiency of directed trees, i.e. arborescences, generated by the implemented heuristic is covered in Voss [14].

3. Steiner tree problem and multicast connections

Problems arising in the domain of routing in communication networks may often be decomposed into a number of well-known combinatorial problems; the minimum Steiner tree problem in graphs, frequently called the Steiner problem, is one of them. The problem of finding a multicast tree topology where minimisation of resource utilisation is of primary concern is analogous to the Steiner tree problem in graphs. Formal definition of the minimum Steiner tree problem can be written as follows:

Definition 1. Let $G = (V, E)$ be an undirected connected graph of the communication network with the nodes set V, the connections set E, and non-negative weights associated with the connections. In this graph we have a set $Q \subseteq V$ of destination nodes, called the multicast group. A Steiner tree problem is to find a minimum cost subgraph of G, such that there exists a path in the subgraph between every pair of destination nodes. In order to realise this subgraph, additional nodes from $V \backslash Q$ may be included.

The optimal solution of the Steiner tree problem is NP-complete [15] and thus not suitable for real-time applications. A number of optimal algorithms for solving the Steiner problem have been developed in the past [1,2]. An important consideration for NP-complete problems is the existence of efficient heuristics with demonstrably good performance [16,14]. Multicast routing algorithms are based on heuristic algorithms.

The minimal information needed by nodes in the network to carry out any sensible routing calculation consists of the distance to each destination, i.e. cost of the minimum cost path, as well as the identity of the first node on the minimum cost path toward the destination. Only a subset of Steiner tree heuristics have the properties that make them suitable for distributed implementation in networks where nodes have minimal routing information. The distributed minimum spanning tree (DMST) algorithm and the distributed Kou–Markowsky–Berman (DKMB) algorithm have been shown in Rugelj and Klavžar [4] to be less efficient in terms of the number of exchanged messages. They are all based on the so-called 'waving' technique that is described in the next section. Regarding cost efficiency, the minimum spanning tree heuristic performs worst, while the cheapest insertion heuristic and the Kou–Markowsky–Berman heuristic are cost comparable [16]. The algorithm in Bauer and Varma [10] is a typical representative of the 'waving' algorithm. It is based on the version of the cheapest insertion heuristic, with additional joining of fragments as several subtrees are built in parallel.

The distributed routing algorithms that solve constrained version of multicast routing problem are not directly comparable with our algorithm although similar techniques of collecting distributed information can be observed. The representative of the 'waving' algorithm is presented in Kompella et al. [17] while the algorithms in Jia et al. [13] and Yonjun-Im and Yanghee-Choi [18] are closer to our table-based approach.

4. Cheapest insertion heuristic

The cheapest insertion heuristic (CI) [19] best suits distributed environments, due to its effectiveness and simplicity. The heuristic is based on a single tree growth. At each step a minimal cost path is added, from the existing partially built tree to the destination node that has not yet been connected. The application of the CI heuristic to a graph with six destination nodes is illustrated in Fig. 1. The randomly picked destination node A represents the initial partially built tree.

The time complexity of the heuristic is $O(|Q||V|^2)$. The multiplier $|V|^2$ comes from the minimum cost path computation between the vertices in the graph.

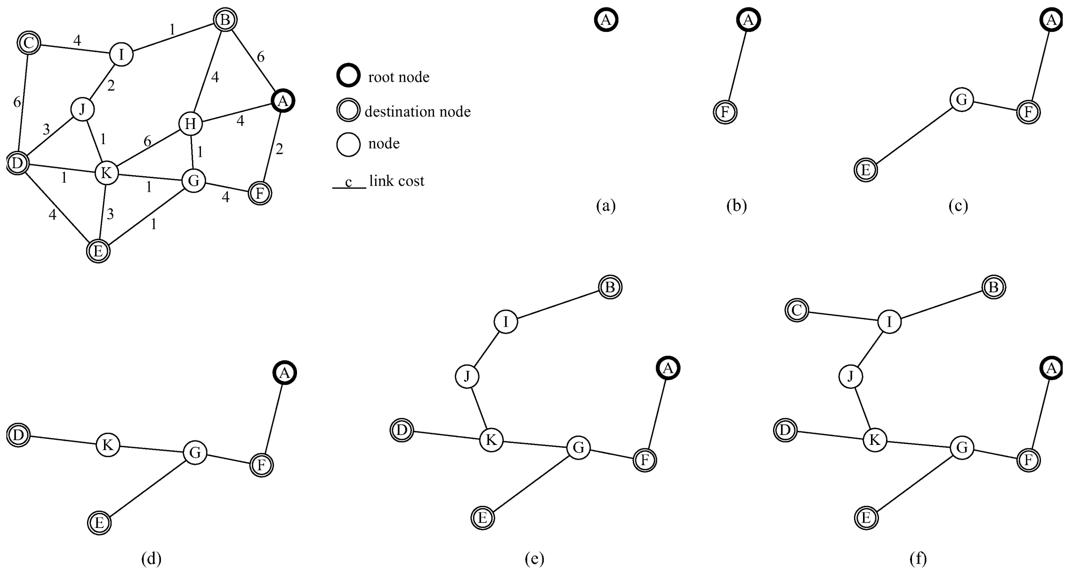

Fig. 1. Application of the CI heuristic to a graph.

5. Distributed CI_W algorithm

The distributed cheapest insertion using waving (CI_W) algorithm [4] implements the CI heuristic in network environments where the centralised information about network topology is not available.

The main problem to be solved in order to implement the CI heuristic in network environments is the selection of the closest destination node to the partially built tree because the minimum cost path information is distributed among network nodes. The authors of the CI_W algorithm proposed a solution based on 'information collecting' and 'decision-distributing' waves generated in the leaf nodes of the growing Steiner tree and propagated towards its root and vice versa. The wave is a sequence of messages sent across the network connections. The node, where the algorithm was initiated, has the role of the tree root. Each tree node selects the closest destination node according to its local information. While the selection wave propagates towards the root, the tree node participates with its candidate in the global distributed decision. The root node makes the final selection. The selection is distributed to the tree nodes by the wave in the opposite direction. The closest node to the selected destination node makes a connection and activates the selected destination node as well as all the intermediate nodes on the path to it.

The execution of the CI_W algorithm in the network with the topology given in Fig. 1 is shown in Fig. 2. The sequence of messages of the type *Select* on the same graph forms the 'information collecting' wave while the sequence of messages of the type *Announce* forms the 'decision distributing' wave. The total number of messages exchanged is 43. The longest chain of sequentially exchanged messages determines the execution time of the algorithm. In our example, the longest chain consists of 34 messages.

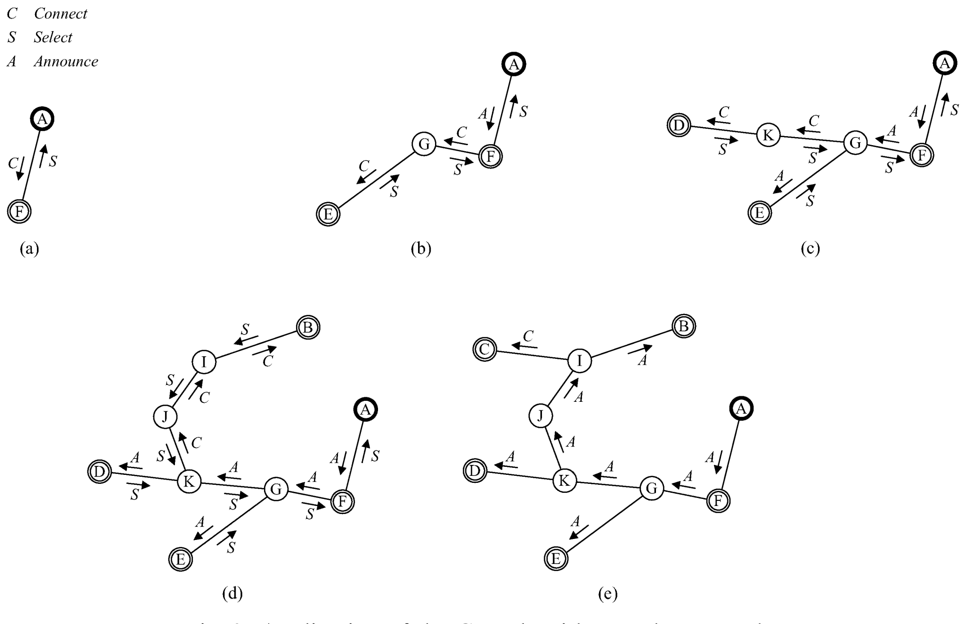

Fig. 2. Application of the CI_W algorithm to the network.

The time complexity of the CI_W algorithm, where the time is measured as the number of sequentially exchanged messages, is bounded by $O(|Q|depth(T))$, where $depth(T)$ denotes the depth of a tree [4]. However, the complexity of the cumulative communication time, i.e. time without considering parallel actions, is limited by $O(|Q||V_T|)$, which is also an upper limit on the number of exchanged messages. $|Q| - 1$ steps are performed with the number of exchanged messages at each step being proportional to $|V_T| - 1$. When the tree nodes consist only of destination nodes, the total number of exchanged messages reaches the lower limit $\Omega(|Q|^2)$.

6. Distributed CI_T algorithm

Further optimisation of the distribution of the CI heuristic along network nodes is possible, assuming that each of them has information only about the minimum cost paths to all other available nodes. The objectives of the optimisation are minimisation of both the time complexity of the algorithm and the amount of the information exchanged between network nodes. Enabling nodes to collect the information needed in advance can reduce the time needed to select the next closest node. In Fig. 3 an example of the inefficiency of the CI_W algorithm is given. Three waves are shown. The root node A selects the closest node E after the selection wave completes (a). The decision is distributed in the next step (b). The inefficiency of the algorithm can be seen at the following selection step (c). A subtree on the left proposes the same closest node as in the previous selection step. This kind of situation will happen whenever the closest node to the subtree is not at the same time the closest node to the whole tree. The overall communication time and the number of exchanged messages can be reduced by an improved algorithm.

The cheapest insertion using table (CI_T) algorithm establishes the topology of multicast connection by means of a distributed table-passing technique. First, the table needs to be defined.

Definition 2. The table is a data structure that keeps a record for each destination node not included in the partially built tree, $Q \setminus V_T$. In each record the identity of the destination node p, the identity of the closest node v, in the partially built tree, to the destination node p, and the cost of the path $c(v, p)$ between these two nodes are stored.

The table consists of the proposed minimum cost paths from the growing tree to each as yet unconnected group member node. The CI_T algorithm can be described non-formally as follows.

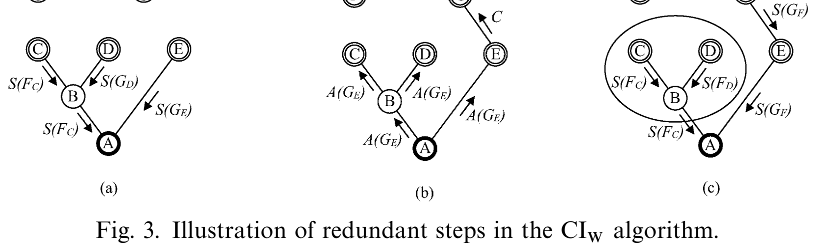
S(X_Y) Select(closest node_originating node)
A(X_Y) Announce(selected node_originating node)
F
G
C
D
E
B
A
S(F_C)
S(G_D)
S(G_E)
(a)
C
A(G_E)
(b)
S(F_D)
S(G_F)
(c)

Fig. 3. Illustration of redundant steps in the CI_W algorithm.

The group member node that was added last decides, according to the information available in the table, which member node should be added next. It then passes the table to the tree node from which the just selected minimum cost path proposition originates. The selected node establishes a connection with the nearest group member node and passes the table through the intermediate nodes to the newly connected node. The intermediate nodes on the newly created path, including the destination node, update the table according to their local information. If the node has a cheaper proposal for a specific unconnected group member node, it replaces the table record for that node. The distributed algorithm terminates its execution when all the member nodes are connected. The formal specification of the $\mathrm{CI_T}$ algorithm in Z-notation is given in the appendix.

The proposed distributed algorithm actually implements the CI heuristic as demonstrated by the following reasoning. After selection of a record that contains the path description between the tree node and one of the destination nodes, the distributed algorithm actually adds the selected path to the tree. This holds as the table is passed to the source of the selected path in the tree and then sent toward destination node of the same path. We need only to prove that the table, after it is updated in a node, contains only records on minimum cost paths between the tree and each unconnected destination node. Initially, the root node creates a table that satisfies this requirement. When the table is later received by a node, which is just added to the tree, only records to those destination nodes are updated for which a cheaper path exists from the current node. When the next closest destination node is selected, its record is removed from the table. Using the principle of induction we can conclude that the updated table in each node satisfies the stated requirement.

Fig. 4 illustrates the application of the $\mathrm{CI_T}$ algorithm to the network in Fig. 1. The table is drawn next to the node where it resides at a given time. Two types of message are used. Messages of the type *Connect* establish a new path from the partially built tree, while messages of the type *Pass* only transfer the table to the next selected node. Initially, the table consists of $|Q| - 1$ records. As the algorithm proceeds, the table gets smaller. The total number of messages exchanged is 12, which is also the algorithm's execution time measured as the number of sequentially exchanged messages.

Although the execution of the $\mathrm{CI_T}$ algorithm is distributed among network nodes no parallel activities are performed. A particular node receives a message, performs a computation and sends a new message to the chosen node. Thus, the number of sequentially exchanged messages is equal to the number of all exchanged messages. The number of exchanged messages of the type *Connect* is $|V_\mathrm{T}| - 1$. Messages of the type *Pass* are sent at most $|Q| - 2$ times between pairs of tree nodes, using the minimum cost paths. A rarely achieved upper bound for the length of these paths is $|V| - 1$. A better approximation would be *depth*(T), which can be achieved by sending the messages of the type *Pass* using the tree connections. In the former case the execution time and the number of exchanged messages are bounded by $O((|Q| - 2)|V| + |V_\mathrm{T}| - 1)$ and in the latter case the execution time by $O((|Q| - 2)\mathit{depth}(\mathrm{T}) + |V_\mathrm{T}| - 1)$ and the number of exchanged messages by $O((|Q| - 1)(|V_\mathrm{T}| - 1))$. Note that these bounds are also bounds for the cumulative communication time. The advantage of the $\mathrm{CI_T}$ algorithm over the $\mathrm{CI_W}$ algorithm is hidden in its average case behaviour and is covered in the next section. The number of messages exchanged on a given path is, in most networks, positively correlated with the cost of the path. Under such conditions the average length of the minimum cost path between pairs of tree nodes is less than *depth*(T).

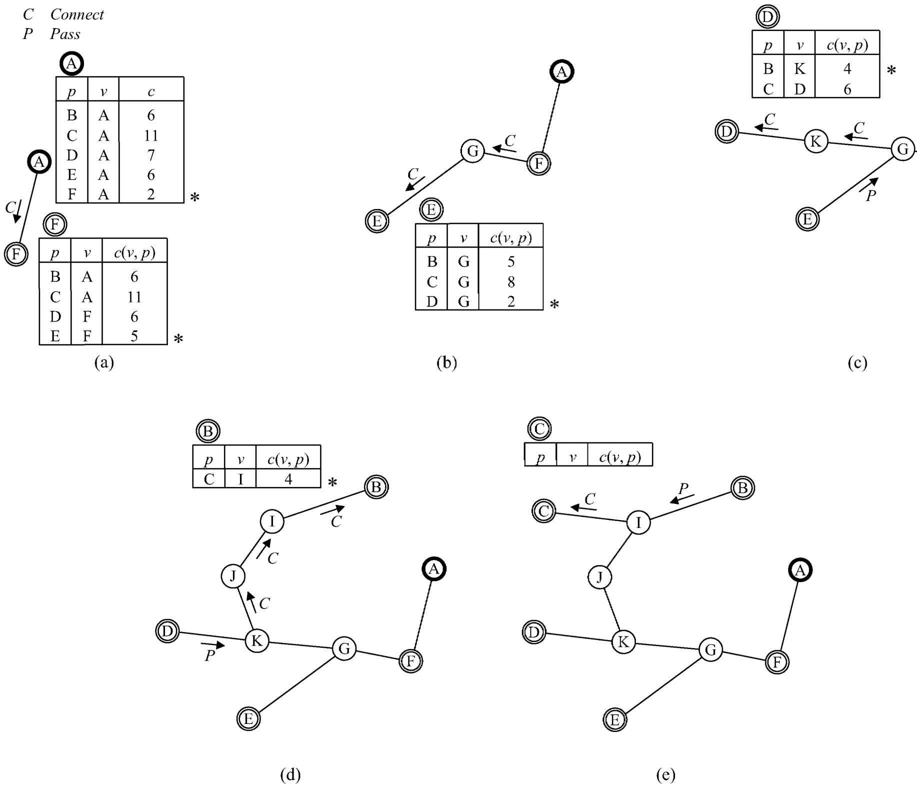

Fig. 4. Application of the CI_T algorithm to the network.

7. Performance measures

The upper bounds on the time complexity and the upper bounds on the number of exchanged messages for both algorithms were given in previous sections. They suggest an improvement in performance of the CI_T algorithm over the CI_W algorithm. However, the main advantage of the CI_T algorithm over the CI_W algorithm is expected in the average case behaviour. The average case performance of the algorithms depends strongly on the topology of the network as well as on the number of the destination nodes, therefore a simulation is required.

7.1. Simulation-based evaluation and validation tool

We have developed a tool for simulation-based distributed algorithm evaluation and carried out simulations on random graphs. In addition to performance evaluation, our tool allows statistical verification of some correctness criteria. We formalise correctness criteria as claims about the behaviour of a distributed algorithm. Three fairly standard correctness criteria are the absence of deadlocks, livelocks, and improper terminations. Other assertions can be checked during simulation runs including system invariants, i.e. a more general case of boolean correctness criteria that, if true in the initial state, remain true in all attainable states. The assertions can refer to all elements of a system state: algorithm variables, control-flow points, and the contents of message queues. An example of an assertion checked is the property of the table that, after it is updated in a node, it contains only records on minimum cost paths to each unconnected destination node.

The core of the tool is a non-preemptive scheduler of node algorithms integrated with management of message queues. Message queues are modelled as infinite length FIFO queues. The basis of time measurement is a logical clock. The number of clock ticks required for a node's processing and inter-node communication can vary and must be configured. The time required for message transfer can be made dependent on the length of the message. In order to measure the length of the longest sequence of exchanged messages a configuration is used where message transfer requires one clock tick while processing at a node consumes no time.

The other important component of our tool is a network topology generator. The parameters for random generation can be selected in such a way that the connectivity characteristics of graphs are very similar to those observed in real networks. For each run a graph is generated according to the network model introduced by Waxman [20] and improved by Doar [21]. Graphs are constructed by distributing n nodes across a Cartesian co-ordinate grid. The location of each node has integer co-ordinates and multiple nodes are not permitted to exist at any location. The connections are added to the graph by considering all possible pairs (u,v) of nodes and using the probability function $P(u,v) = ke\exp(-d(u,v)/(\alpha L))/|V|$ where e is the mean degree of the node, $|V|$ is the cardinality of V in G, $d(u,v)$ is the Euclidean distance between the node locations, L is the maximum possible distance between two nodes, α is a parameter in the range $0 < \alpha \leqslant 1$, and k is a scale factor related to the mean distance between two random nodes. A large value of α increases the number of connections to nodes further away, while a large value of k increases the number of connections from each node. The cost of the connection is defined as $d(u,v)$. Each graph is tested to ensure that only one connected component exists.

7.2. Results

The results presented were obtained using network topologies with $\alpha = 0.25$ and $k = 3.5$. Results are shown for topologies where the nodes had, on average, 5 and 10 neighbouring nodes. The size of the network as well as the size of the multicast group was up to 60 nodes. For each set of simulation parameters 2000 simulations were performed. The following figures show average values and standard deviations of performance measures under consideration.

Simulation results have confirmed the superiority of the $\mathrm{CI_T}$ algorithm over the algorithm based on waving technique, as regards all performance measures: the average execution time, the average number of the exchanged messages, and the average amount of the exchanged information. We were not interested in the cost of the resulting tree. The cost efficiency of the CI heuristic has been shown previously [16,14].

The average time measured as the number of sequentially exchanged messages required by the algorithms in a graph with $|V| = 60$ and $e = 5$ is shown in Fig. 5 as a function of the number of destination nodes. The $\mathrm{CI_T}$ algorithm is significantly faster when the number of the destination nodes is high. The size of a network influences the execution time of both algorithms. Fig. 6 shows the increase in execution time for a group of 15 destination nodes.

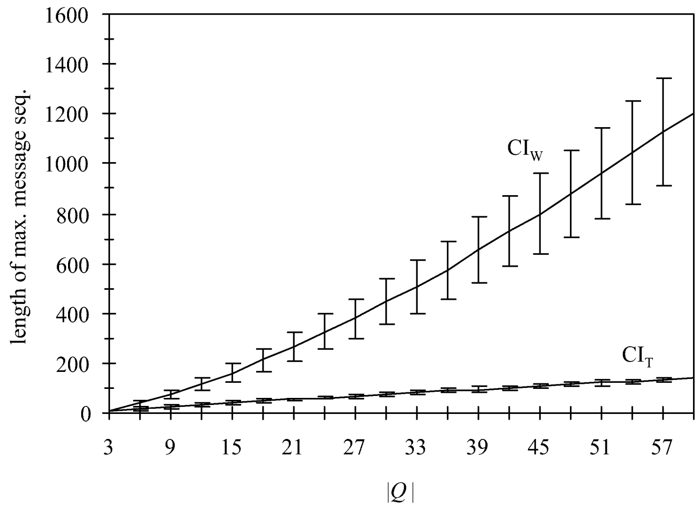

Fig. 5. Average execution time as a function of $|Q|$ ($|V| = 60$, $e = 5$, $\alpha = 0.25$, $k = 3.5$).

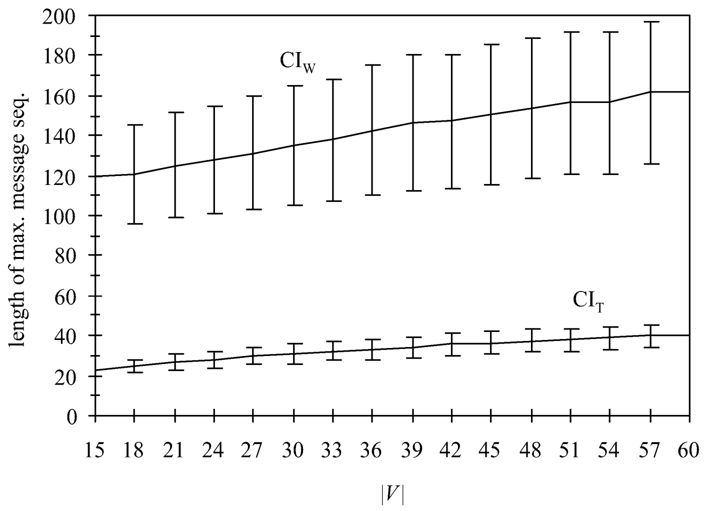

Fig. 6. Average execution time as a function of $|V|$ ($|Q| = 15$, $e = 5$, $\alpha = 0.25$, $k = 3.5$).

The CI_T algorithm is slightly faster in dense networks, even when each message must be chunked into several messages, as it is the case in ATM networks. In Figs. 7 and 8, in addition to execution time of both algorithms in dense networks, the results of simulation runs where the table was filled in 48 bytes long messages (CI_{T48}) are also shown.

The average number of exchanged messages is a measure of the average cumulative communication time, i.e. time without considering parallel actions. With increasing group size, the average number of exchanged messages increases steeply using the CI_W algorithm. That is not the case

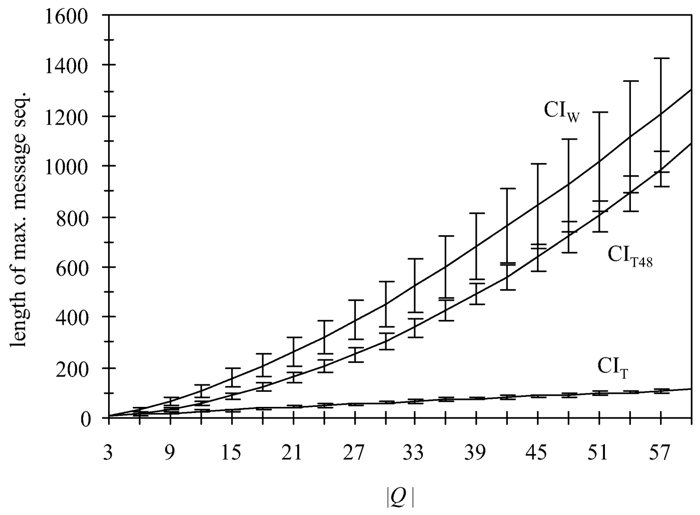

Fig. 7. Average execution time as a function of $|Q|$ in dense networks ($|V| = 60$, $e = 10$, $\alpha = 0.25$, $k = 3.5$).

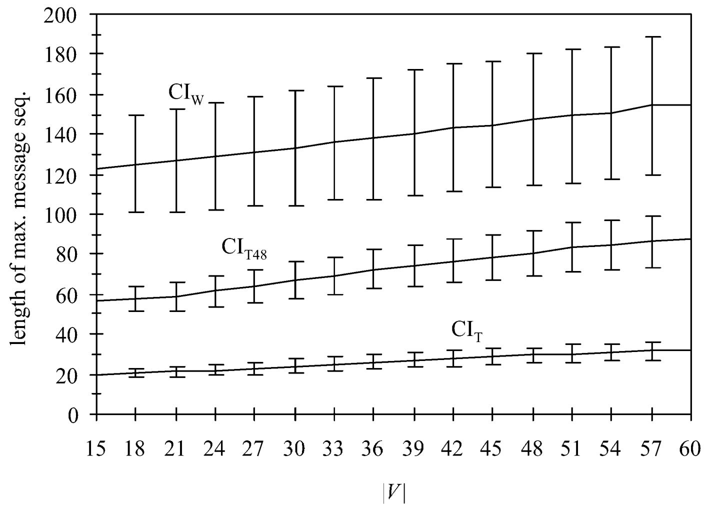

Fig. 8. Average execution time as a function of $|V|$ in dense networks ($|Q| = 15$, $e = 10$, $\alpha = 0.25$, $k = 3.5$).

when the CI_T algorithm is used, as can be seen in Fig. 9. Fig. 10 shows the number of exchanged messages as a function of $|V|$. The efficiency of the table-passing technique is based on the fact that the table contains information for future decisions.

The comparison of exchanged information is more appropriate for some type of networks as the table in the CI_T algorithm contains several fixed size records, while messages in the CI_W algorithm contain only a single fixed size record, with the exception of $|V_T| - 1$ messages that contains sets of destination nodes. The total amount of exchanged information for both algorithms is shown as

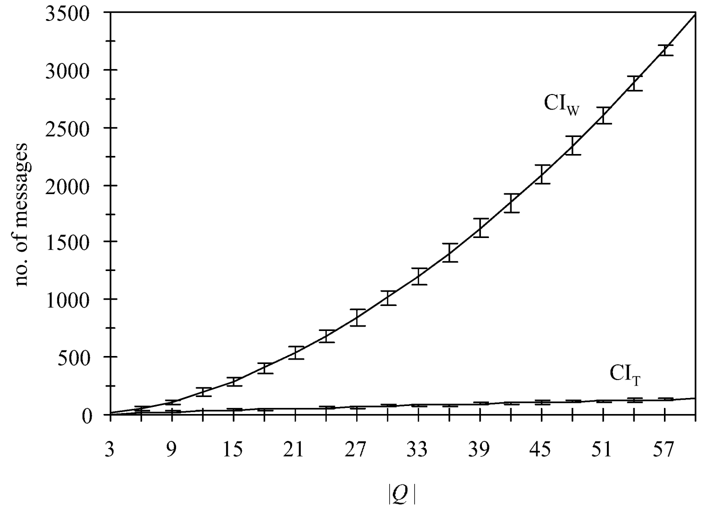

Fig. 9. Average number of exchanged messages in a 60 node network as a function of $|Q|$ ($|V| = 60$, $e = 5$, $\alpha = 0.25$, $k = 3.5$).

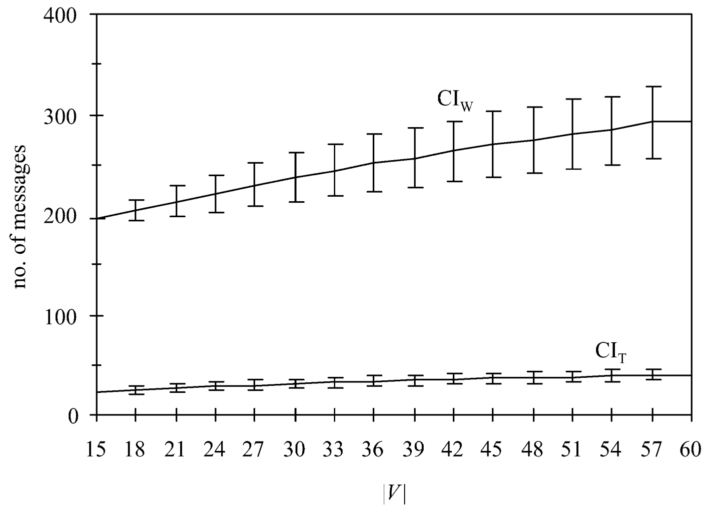

Fig. 10. Average number of exchanged messages as a function of $|V|$ ($|Q| = 15$, $e = 5$, $\alpha = 0.25$, $k = 3.5$).

a function of $|Q|$ (Fig. 11) and as a function of $|V|$ (Fig. 12). It can be clearly seen that redundant data is present in the messages exchanged by the $\mathrm{CI_W}$ algorithm.

Appendix. The $\mathrm{CI_T}$ algorithm

A formal specification of the $\mathrm{CI_T}$ algorithm is given in the Z specification notation [22,23] (Fig. 13). The table records are represented as a partial function *Table* that returns the couple (v, c)

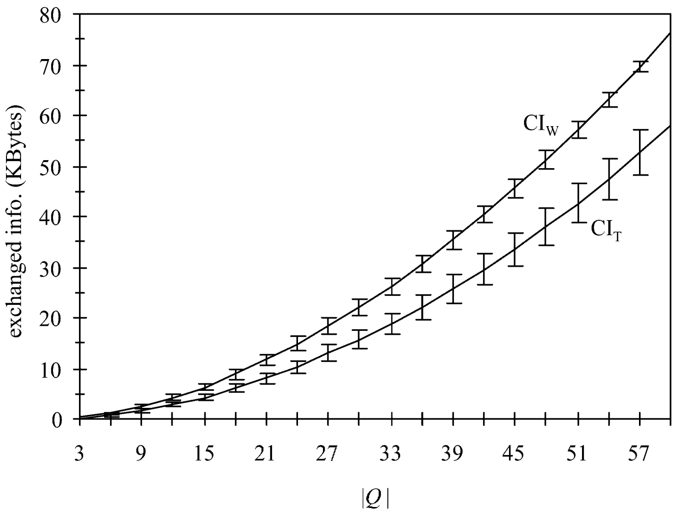

Fig. 11. Average amount of exchanged information as a function of $|Q|$ ($|V| = 60$, $e = 5$, $\alpha = 0.25$, $k = 3.5$).

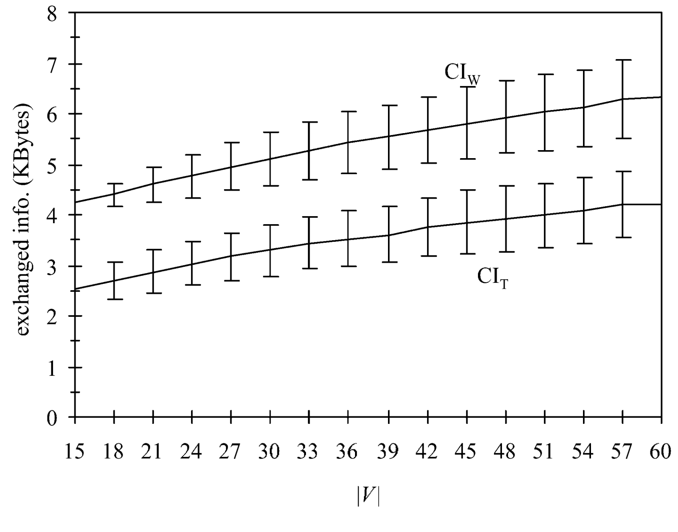

Fig. 12. Average amount of exchanged information as a function of $|V|$ ($|Q| = 15$, $e = 5$, $\alpha = 0.25$, $k = 3.5$).

for a given unconnected destination node *p*. A variable *self* contains the address of the node in which the algorithm is being executed. The function *costTo* returns the cost of the minimum cost path and the function *nextTo* returns the identity of the first node on the minimum cost path to a given node. The variable *connection* contains a set of immediate neighbours in the established connection for given node. The sequences *input* and *output* stand for the node's input and output

```
$[Address]$
$Table == Address \pfun (Address \times \mathbb{N}_1)$
$MsgType ::= Connect \mid Pass$

| $costTo : Address \pfun \mathbb{N}_1$
| $nextTo : Address \pfun Address$
| $self : Address$

┌─ $Memory$
│ $connection : \mathbb{P}\ Address$
│ $input, output : \mathrm{seq}\ (Address \times Address \times MsgType \times Table)$
└──

| $select : Table \rightarrow Address$
|──
| $\forall\ tab : Table \bullet$
|     $(\exists\ dest : \mathrm{dom}\ tab \bullet$
|         $(\forall\ q : \mathrm{dom}\ tab \bullet second(tab(q)) \geq second(tab(dest))) \wedge select(tab) = dest)$

| $update : Table \rightarrow Table$
|──
| $\forall\ tab : Table \bullet$
|     $update(tab) = tab \oplus \{dest : \mathrm{dom}\ tab \mid second(tab(dest)) > costTo(dest) \bullet$
|         $dest \mapsto (self, costTo(dest))\}$

┌─ $InitRoot$
│ $\Delta Memory$
│ $Q : \mathbb{P}\ Address$
├──
│ $Q \setminus \{self\} \neq \varnothing$
│ $\exists\ tab : Table; dest : Address \mid$
│     $tab = \{q : Q \setminus \{self\} \bullet q \mapsto (self, costTo(q))\} \wedge dest = select(tab) \bullet$
│     $(connection' = \{nextTo(dest)\} \wedge$
│     $output' = output \frown \langle(nextTo(dest), dest, Connect, tab)\rangle)$
└──

$ConnectEvent \mathrel{\widehat{=}} ForwardConnect \vee Decide$

┌─ $ForwardConnect$
│ $\Delta Memory$
├──
│ $\exists\ from, dest : Address; tab : Table \mid$
│     $(from, dest, Connect, tab) = head(input) \wedge dest \neq self \bullet$
│     $(connection' = \{from, nextTo(dest)\} \wedge input' = tail(input) \wedge$
│     $output' = output \frown \langle(nextTo(dest), dest, Connect, update(tab))\rangle)$
└──

┌─ $Decide$
│ $\Delta Memory$
├──
│ $\exists\ from : Address; tab, ntab : Table \mid$
│     $(from, self, Connect, tab) = head(input) \wedge ntab = update(\{self\} \lhd tab) \bullet$
│     $(connection' = \{from\} \wedge input' = tail(input) \wedge$
│     $(\mathrm{dom}\ ntab \neq \varnothing \Rightarrow$
│         $(\exists\ dest : Address \mid dest = first(tab(select(ntab))) \bullet$
│         $output' = output \frown \langle(nextTo(dest), dest, Pass, ntab)\rangle)))$
└──

┌─ $StartConnect$
│ $\Delta Memory$
├──
│ $\exists\ from, dest : Address; tab : Table \mid$
│     $(from, self, Pass, tab) = head(input) \wedge dest = select(tab) \bullet$
│     $(input' = tail(input) \wedge connection' = connection \cup \{nextTo(dest)\} \wedge$
│     $output' = output \frown \langle(nextTo(dest), dest, Connect, tab)\rangle)$
└──
```

Fig. 13. The CI_T algorithm.

queues. Receive and send actions are achieved respectively by extracting the first ordered tuple from the sequence and by adding a new ordered tuple to the sequence. The global function *select* returns the destination node for which the minimum cost record exists. The global function *update* performs an update of a given table with respect to the node's local information.

The initiating node is the destination node that is presented with the set of destination nodes. The schema *ConnectEvent* describes the response of the node to the receipt of the *Connect* message. The distinction is made between intermediate nodes and destination node. The topology of the multicast connection is established when the table is empty. The message of the type *Pass* is forwarded to the final destination *dest*, where it is processed in accordance with the schema *StartConnect*.

References

[1] Winter P. Steiner problem in networks: a survey. Networks 1987;17:129–67.
[2] Hwang FK, Richards DS. Steiner tree problems. Networks 1992;22:55–89.
[3] Jaffe JM. Distributed multi-destination routing: the constraints of local information. SIAM Journal of Computing 1985;14:875–88.
[4] Rugelj J, Klavžar S. Distributed multicast routing in point-to-point networks. Computers and Operations Research 1997;24:521–7.
[5] Rugelj J. Distributed multicast routing mechanism for global point-to-point networks. Proceedings of the 20th Euromicro Conference, Liverpool, UK, 1994. p. 388–95.
[6] Novak R, Kandus G. Adaptive Steiner tree balancing in distributed algorithm for multicast connection setup. Microprocessing and Microprogramming 1994;40:795–8.
[7] Diot C, Dabbous W, Crowcroft J. Multipoint communication: a survey of protocols, functions, and mechanisms. IEEE Journal on Selected Areas in Communications 1997;15:277–90.
[8] Deering S, Estrin D, Farinacci D, Jacobson V, Liu CG, Wei L. The PIM architecture for wide-area multicast routing. IEEE/ACM Transactions on Networking 1996;4:153–62.
[9] Gelenbe E, Ghanwani A, Srinivasan V. Improved neural heuristics for multicast routing. IEEE Journal on Selected Areas in Communications 1997;15:147–50.
[10] Bauer F, Varma A. Distributed algorithms for multicast path setup in data networks. IEEE/ACM Transactions on Networking 1996;4:181–91.
[11] Shaikh A. Destination-driven routing for low-cost multicast. IEEE Journal on Selected Areas in Communications 1997;15:373–81.
[12] Sriram R, Manimaran G, Siva Ram Murthy C. Algorithms for delay-constrained low-cost multicast tree construction. Computer Communications 1998;21:1693–706.
[13] Jia X, Zhang Y, Pissinou N, Makki K. A distributed multicast routing protocol for real-time multicast applications. Computer Networks 1999;31:101–10.
[14] Voss S. Worst-case performance of some heuristics for Steiner's problem in directed graphs. Information Processing Letters 1993;48:99–105.
[15] Karp RM. Reducibility among combinatorial problems. In: Complexity of computer computations. New York: Plenum Press, 1972. p. 85–104.
[16] Voss S. Steiner's problem in Graphs: Heuristic methods. Discrete Applied Mathematics 1992;40:45–72.
[17] Kompella VP, Pasquale JC, Polyzos GC. Optimal multicast routing with quality of service constraints. Journal of Network and Systems Management 1996;4:107–31.
[18] Yonjun-Im, Yanghee-Choi. A distributed multicast routing algorithm for delay-sensitive applications. Proceedings of ICPADS'98 – International Conference on Parallel and Distributed Systems, 1998. p. 232–9.
[19] Takahashi H, Matsuyama A. An approximate solution for the Steiner problem in graphs. Math. Japon. 1980;24:573–7.

[20] Waxman BM. Routing of multipoint connections. IEEE Journal on Selected Areas in Communications 1988;6:1617–22.

[21] Doar M, Leslie I. How bad is naive multicast routing. Proceedings of IEEE INFOCOM'93, 1993. p. 82–9.

[22] Spivey JM. The Z notation. Prentice-Hall international series in computer science. New York: Prentice-Hall, 1992.

[23] Jacky J. The way of Z: practical programming with formal methods. Cambridge: Cambridge University Press, 1996.