Reliable Overlay Multicast Trees for Private Internet Broadcasting with Multiple Sessions

Chae Y. Lee and Ho Dong Kim

Dept. of Industrial Engineering, KAIST,

373-1 Kusung Dong, Taejon, Korea

{chae, hdkim}@kaist.ac.kr

Abstract

With rapid advances of computing technologies and high speed networks, various high volume multimedia services have become popular in the Internet. Private Internet broadcasting is a typical way to support these services and overlay multicast technology is known to be a promising solution to support this method. In an overlay multicast network, members are dynamically joining or leaving their multicast group. To reduce frequent updates of multicast members and provide a reliable multicast route, overlay multicast trees are investigated. The problem is formulated as a binary integer programming which maximizes the minimum link reliability for all multicast sessions. Tabu search heuristic is developed with repeated intensification and diversification. Robust computational result is obtained that is comparable to the optimal solution and applicable in real time.

Keywords

Overlay multicast, Private Internet broadcasting, Multicast session tree, Link reliability, Tabu search

1. Introduction

Various high volume multimedia services such as electronic newspapers, video-conferencing, video-on-demand (VOD) and cyber education have become popular with rapid advances of computing technologies and high speed networks. These multimedia contents are transmitted over the Internet either in download mode or in streaming mode (i.e., video streaming and audio streaming) [1]. In the download mode, a service user can download the whole multimedia file and then play back the file in his computer. Streaming mode is necessary for Internet broadcasting, since video contents need not be downloaded in full and is being played out while parts of the file are received. We in this paper are interested in video streaming service for private Internet broadcasting [2, 3, 4].

For multimedia services, broadcasting is gradually magnified by contents suppliers, service providers and receivers. If users have basic computer equipments such as image camera, multimedia software and sound card, they can provide live Internet broadcasting. This private Internet broadcasting [2] is different from commercial one in that its service time period is restricted. A typical example of private Internet broadcasting is SHOUTcast [5] where a distributed streaming audio service is provided.

For private Internet broadcast, multicast transmission technology has been suggested which has network bandwidth efficiency [6]. Differently from unicast, multicast permits a source to get away with transmitting only single stream, regardless of how many receivers exist in its multicast group. One stream of video is sent to a multicast router and the router sends the packet to one segment of the network on which there is a receiver. Traditional IP multicast which runs at network layer is not widely deployed because of technical and economical reasons [7, 8]. Difficulty of inter-domain communications, scalability problem and additional software requirements on routers hindered the deployment of IP multicast. Overlay multicast is an emerging technology appearing as a break-through to solve these problems. Overlay multicast, which runs at the transport layer, is a scalable unicast-based multicast technique among group members. Since overlay multicast resides on top of densely connected IP network, it can be easily deployed by effectively reducing complexity of network routers.

One important issue in overlay multicast is to build a multicast tree, where video streams are transmitted from a source node to its members through the path. Algorithms suggested for overlay multicast tree construction are classified into two branches [9]: Centralized and distributed algorithms. Centralized algorithms assume that all multicast members know the network information. Distributed algorithms, on the other hand, assume that a member node knows only partial information of adjacent nodes and links. Typical examples of the centralized algorithm are ALMI [10] and Host-Based Multicast (HBM) [11]. ALMI is designed for a large number of groups, each with a small number of members. The objective of ALMI is to minimize the cost of multicast trees where the cost of each link is the round-trip delay. HBM is proposed to fast recover network failures, when multicast members fail to forward packets.

Decentralized algorithms are divided into tree-first and mesh-first approaches. In the tree-first approach, a multicast tree can easily be reconstructed when a new node arrives or an existing node fails. In the mesh-first approach, all multicast members maintain a connected mesh topology and multicast tree is built up on top of it. The tree-first approach includes YOID [12], Overcast [13] and NICE [14] and the mesh-first includes Narada [15] and Scattercast [16]. YOID intends to improve delay and data loss rate in the situation where a tree changes dynamically. Overcast is designed for scalable and reliable single-source multicast trees under changing network conditions. NICE intends to produce a number of different trees to improve quality of data streams. The aim of Narada and Scattercast is to minimize the delay from a multicast source to each multicast member.

In this paper, we are interested in designing reliable overlay multicast trees with multiple sessions. These multiple sessions are serviced by single source. A typical example of the multiple sessions is lecture webcasting [17]. In the webcasting, one video streaming shows the speaker and a second streaming shows the presentation material. Each session of video streaming service has a group of members whose join and leave actions are centrally controlled by the source. Our objective is to build a reliable multicast tree for each session that satisfies some common constraints of an overlay network, which includes the end-to-end delay bound.

The rest of this paper is structured as follows. In Section 2, we discuss overlay multicast for private Internet broadcasting, where restrictions in the design of overlay multicast trees are examined.

Section 3 presents formulation of reliable overlay multicast trees with multiple sessions. Tabu search heuristic is developed to solve the problem in Section 4. Construction of initial solutions and intensification and diversification processes are examined. Computational results and conclusion are provided in Section 5 and 6 respectively.

2. Overlay Multicast for Private Internet Broadcasting

Multimedia streaming services are now at the beginning of operation over the Internet. Commercial streaming networks such as Akamai Technologies [18] and Real Network [19] have deployed large-scale live streaming infrastructures.

As shown in Figure 1, a source of Internet broadcasting produces live audio or video streaming to provide service to a group of receivers. Due to high bandwidth requirement of video streaming, it is unusual to transmit raw video streaming over current Internet environment. Thus, encoders are used to compress raw streaming data before transmitting them to receivers. A receiver decodes audio and video streaming and enjoys broadcasting services. The receiver also sends its statistical file to the source to inform the time of joining or leaving the service, service rate and other user request.

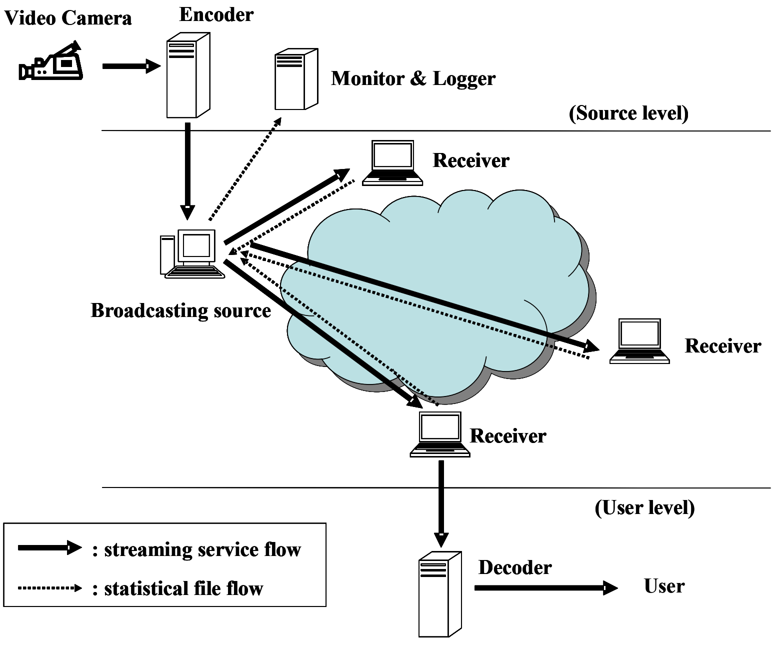


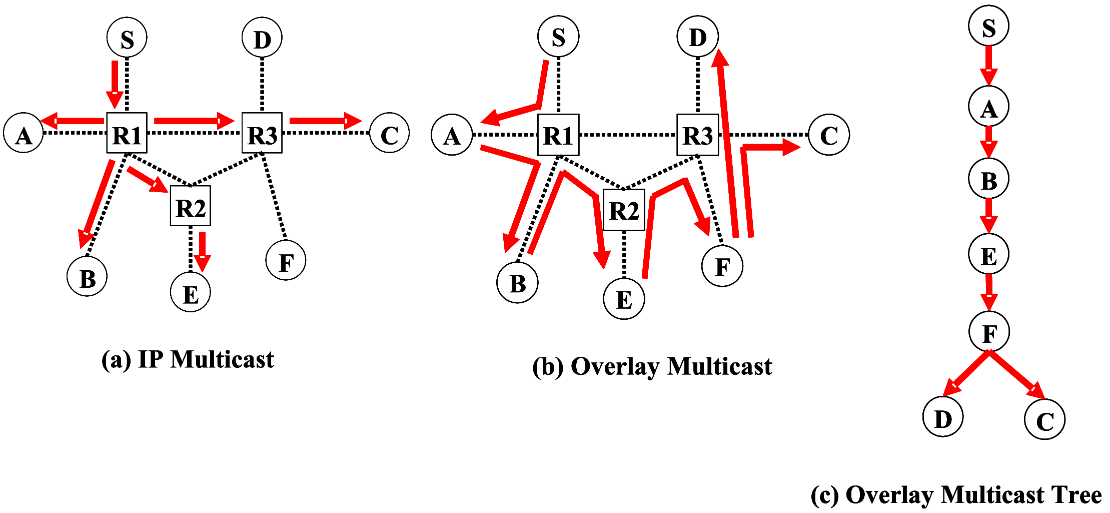
Figure 1. Overview of private Internet broadcasting system

Table 1. A log file format for streaming service

Field name	Description
remote_host	IP address or Domain Name System (DNS) hostname
auth_user	Name of authenticated user
Time	Time of joining or leaving the service
Event	User request, i.e., command name and targeted service class name
service_rate	record of the service rate per second

In order for the source to provide a service, a log file [20, 21] is essential to analyze the characteristic of each subscriber. Since the log file in [20, 21] is designed for web page service, it may not be suitable for streaming service. Thus, a modified log format is necessary. It must include information of log-in, log-out, and service rate of each member in each session. A suitable log file format for streaming service is recommended as in Table 1. The file includes remote_host, auth_ user, time, event and service_rate. Selection between IP address and DNS hostname for remote_host is decided by the overlay multicast administration. Time of joining or leaving the service is recorded in the field of time with the format of [dd/mm/yyyy:hh:mm:ss]. Thus, accumulated log file can be used as a database for each user.

Now, in overlay multicast, all members including multicast source are connected in application layer. Therefore, member nodes can communicate each other without additional function of routers. Figure 2 shows difference between IP and overlay multicasts. In IP multicast, source node first transmits packets to an adjacent router. The router then copies the data and sends them to other routers and receivers. However, in overlay multicast, the source node transmits packets to adjacent receivers. Receivers then forward them to other receivers. In this way receivers successively forward packets to other members without the aid of routers. IP multicast thus generates unnecessarily many data flows compared to the overlay multicast. Figure 2 (c) shows logical links in overlay multicast tree without underlying physical network. Each link in overlay multicast tree represents an independent unicast session.



(a) IP Multicast

(b) Overlay Multicast

(c) Overlay Multicast Tree

Figure 2. IP and Overlay Multicasts

In overlay multicast tree, each member node receives a data flow and forwards it to one or more successive members. Since each member node has different CPU power and storage size, different performance may result depending on the capacity limit of a member. Thus, we need to consider degree constraint which counts the maximum number of unicast data flows a node can transmit. The average node degree which is mainly dependent on the computing power and buffer size is between three and four [22].

Another issue in designing an overlay multicast tree is the link capacity. A logical link in the tree can traverse one or more physical links in the Internet. It is generally assumed that the shortest physical routing path between member nodes is selected for a logical link. Thus, the link capacity is restricted by the minimum capacity of physical links in the path. When multiple sessions are routed in a logical link, the link capacity is critical in designing trees for those sessions.

In designing an overlay multicast tree, we finally need to consider the joining or leaving of group members. When a member node joins or leaves a multicast session, it needs to be connected into or disconnected from its tree. Connection of a member to a corresponding multicast session tree can be performed by periodically updating the tree. However, updating procedure is not simple when a node leaves a session. If a member leaves a multicast session, all descendant nodes of the member are disconnected. Therefore, we need to design a reliable multicast tree for each session such that the multicast tree is sustainable during the session without frequent updates.

3. Overlay Multicast Trees for Multiple Sessions

An overlay multicast network can be modeled with a directed graph $G = (V, E)$, where V and E represents a set of overlay nodes and links respectively. Given the graph G, we consider K multicast sessions such that each session k has a set of multicast members, represented as S_k, where $S_k \subset V$, $k = 1,...,K$. All sessions are assumed to have same source s. That is, single source provides multiple streaming services in the overlay network. We are interested in constructing an overlay multicast tree for each session that satisfies constraints in the network. For traditional IP multicast routing, Oliveira and Pardalos [23] provide diverse applications of combinatorial optimization.

For a multicast member node $m \in S_k$, a path is required to connect node m to multicast source s. Let x_{ij}^k be a binary variable for link (i, j) in multicast session k. If there is a direct link from node i to j in multicast session k, $x_{ij}^k = 1$. Otherwise, $x_{ij}^k = 0$. Also, let y_{ijm}^k be a binary variable to represent a path between the source and multicast member m of session k. $y_{ijm}^k = 1$, if there is a direct link from node i to j on the path. Otherwise, $y_{ijm}^k = 0$. Then the following relationships hold for flow conservation [24].

$$\sum_{(i,j)\in E} y_{ijm}^k - \sum_{(j,i)\in E} y_{jim}^k = \begin{cases} +1, & \mathit{if}\ i = s,\ \mathit{for\ all}\ k\ \mathit{and}\ i, j, m \in S_k \\ -1, & \mathit{if}\ i = m,\ \mathit{for\ all}\ k\ \mathit{and}\ i, j, m \in S_k \\ 0, & \mathit{otherwise} \end{cases}$$

For a link (i, j) to be inserted in the path from the source to node m of multicast tree k, the link has to be selected for the session as in the constraint below.

$$y_{ijm}^k \le x_{ij}^k, \qquad \mathit{for\ all}\ k, (i, j) \in E\ \mathit{and}\ \ i, j, m \in S_k$$

Since a multicast tree for session k is a spanning tree with n_k members, we have

$$\sum_{(i,j)\in E} x_{ij}^k = n_k - 1, \quad \mathit{for\ all}\ k\ \mathit{and}\ i, j \in S_k$$

Now, each member node in the overlay network has capacity limit represented by degree constraint.

Let D_i be the degree constraint of node i, then we have

$$\sum_{k=1}^{K}\left(\sum_{(i,j)\in E} x_{ij}^k + \sum_{(j,i)\in E} x_{ji}^k\right) \le D_i, \quad \textit{for all } k \textit{ and } i, j \in S_k$$

To satisfy the end-to-end delay bound for members of session k, the following constraint is necessary

$$\sum_{(i,j)\in E} d_{ij}^k y_{ijm}^k \le L_k, \quad \textit{for all } k \textit{ and } i, j, m \in S_k$$

where d_{ij}^k is the delay in link (i,j) and L_k is the delay bound of session k. If $d_{ij}^k = 1$ for all (i,j), L_k becomes the end-to-end hop counts in session k.

In an overlay network with two or more sessions, trees may traverse the same link. These multiple sessions have to satisfy the link capacity. Let r_k be the service data rate of session k and C_{ij} be the capacity of link (i,j). Then link capacity constraint becomes

$$\sum_{k=1}^{K} r_k x_{ij}^k \le C_{ij}, \quad \textit{for all } (i,j) \in E \textit{ and } i, j \in S_k$$

Now, our objective is to have reliable overlay multicast trees that satisfy all constraints discussed above. Here, we assume each node has a sojourn probability that resides in its multicast group during a fixed period of time. After the interval, each multicast session tree is updated with join and leave information of members. If the sojourn probability is high, it is more desirable for other descendant members to be connected to the node. This is because descendant nodes connected to a node with higher sojourn probability may be serviced more reliably during the period. Let p_i^k be the sojourn probability of node i in session k. To represent the link reliability, we adopt a logarithmic function which is usually employed for various utilities. In this study, by applying weight w^k to each session, the link reliability of session k is represented as $w^k \log M p_i^k p_j^k$. Note that the link reliability is distinct for each session due to the sojourn probabilities p_i^k. M is employed to prevent the negative link reliability. When two nodes are not connected, the link reliability is given by $w^k \log M$. By assuming M is as large as desired, the link reliability is consistent to our objective which maximizes the minimum link reliability in all sessions. Thus, the objective function is given by

$$\underset{1\le k\le K}{Max}\left\{\underset{\substack{(i,j)\in E\\ i,j\in S_k}}{Min}\quad w^k\left[x_{ij}^k \log Mp_i^k p_j^k + (1-x_{ij}^k)\log M\right]\right\}$$

According to the above discussion, the design of reliable overlay multicast trees can be formulated as the following binary integer programming problem.

$$\underset{1\le k\le K}{Max}\left\{\underset{\substack{(i,j)\in E\\ i,j\in S_k}}{Min}\quad w^k\left[x_{ij}^k \log Mp_i^k p_j^k + (1-x_{ij}^k)\log M\right]\right\} \tag{1}$$

Subject to:

$$\sum_{(i,j)\in E} y_{ijm}^k - \sum_{(j,i)\in E} y_{jim}^k = \begin{cases} +1, & \text{if } i = s, \text{ for all } k \text{ and } i,j,m \in S_k \\ -1, & \text{if } i = m, \text{ for all } k \text{ and } i,j,m \in S_k \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$$y_{ijm}^k \le x_{ij}^k, \qquad \text{for all } k, (i,j)\in E \text{ and } i,j,m \in S_k \tag{3}$$

$$\sum_{(i,j)\in E} x_{ij}^k = n_k - 1, \qquad \text{for all } k \text{ and } i,j \in S_k \tag{4}$$

$$\sum_{k=1}^{K}\left(\sum_{(i,j)\in E} x_{ij}^k + \sum_{(j,i)\in E} x_{ji}^k\right) \le D_i, \qquad \text{for all } k \text{ and } i,j \in S_k \tag{5}$$

$$\sum_{(i,j)\in E} d_{ij}^k y_{ijm}^k \le L_k, \qquad \text{for all } k \text{ and } i,j,m \in S_k \tag{6}$$

$$\sum_{k=1}^{K} r_k x_{ij}^k \le C_{ij}, \qquad \text{for all } (i,j)\in E \text{ and } i,j \in S_k \tag{7}$$

$$x_{ij}^k,\ y_{ijm}^k \in \{0,1\} \qquad \text{for all } k, (i,j)\in E \text{ and } i,j,m \in S_k \tag{8}$$

Note that the well-known degree constrained or delay constrained spanning tree problem which is a special case of above binary integer programming is NP-complete [25]. This implies that any known exact algorithm will run in time exponential in the size of problem instance. Such an algorithm is thus in most cases unusable for real-world size problems. As encouraging results on NP-complete problems, we investigate a tabu search heuristic to have reliable overlay multicast trees. The tabu search to rearrange the multicast trees will efficiently handle members joining and leaving their multicast groups.

4. Tabu Search

Tabu search [26, 27] is a meta-heuristic procedure for solving optimization problems. It is designed to guide other methods to overcome the trap of local optimality. The main concepts of tabu search includes: 1) tabu lists and tabu list size, 2) tabu restrictions and aspiration criteria and 3) intensification and diversification strategies. In this study the following three steps are considered to obtain the reliable overlay multicast trees as shown in Figure 3.

1) Initial solution (Initial Overlay Multicast Trees)

2) Intensification with a Short-Term Memory

3) Diversification with a Long-Term Memory

The role of a short-term memory is to prohibit moves from recently visited solutions in the intensification process. Recently visited solutions are stored in a tabu list and forbidden to cycle. Since the short-term memory may fail to discover good solutions, a long-term memory is introduced. The long-term memory is to diversify the search space by finding a new solution. Thus, diversification guarantees a global solution while intensification strategies provide an elite solution in a restricted search space.

4.1 Initial Overlay Multicast Trees

As an initial solution for the overlay multicast tree, we need a spanning tree for each session that satisfies node degree, link capacity and delay bound. To obtain an initial overlay multicast tree, two strategies are considered; “Reliable Initial Tree” and “Random Initial Tree”. For the reliable initial tree, links are sequenced in nonincreasing order of reliability in each session. Then, a spanning tree is built by selecting links in the order of reliability. Since the node degree and link capacity are commonly enjoyed by all sessions, links that satisfy two constraints are selected by each session. A tree constructed by the procedure above may not satisfy delay bound. To have a feasible solution, a node in each session that does not satisfy the delay bound is selected. The node and its descendants are reconnected to an ascendant node of the tree which satisfies the delay bound with higher link reliability. The reconnection process is continued until all descendant nodes satisfy the delay bound. Random initial tree is built by randomly selecting links that satisfy node degrees and link capacities. To have a feasible solution, a node that does not satisfy the delay bound is reconnected to an ascendant node of the tree. This reconnection process is continued until all nodes satisfy the delay bound.

4.2 Intensification with Short-Term Memory

Since we have an initial feasible tree for each overlay multicast session, we are now interested in improving the solution by applying tabu search. To have more reliable multicast trees, link swap and link reconnection moves are considered.

In the link swap move, the link with lowest reliability in all sessions is selected as the target link. To increase the reliability of the tree target link (i, j) is exchanged with the highest reliable link (i', j') that satisfies $p_{i'}^k > p_i^k$ and $p_{j'}^k \geq p_j^k$ for links with $p_i^k \geq p_j^k$ and $p_{i'}^k \geq p_{j'}^k$. Thus, swap move which results in (i, j') and (i', j) as in Figure 4 guarantees improved minimum reliability. Link reconnection is simply to connect the node with lower sojourn probability of the target link to other node that best improves the link reliability. Figure 5 shows link reconnection process. Node j with lower sojourn probability of target link (i, j) is reconnected to other node i' which has higher sojourn probability in link (i', j').

Intensification procedure is based on a short-term memory which systematically controls the tabu list. After applying the link swap or link reconnection move, the node with lower sojourn probability of the target link is added to the tabu list with its session. Nodes added into the tabu list are forbidden for a certain period to be reselected as a target node. After each move, frequency count of the newly added link is increased by one, which is a valuable information for diversification. Intensification process is continued until no solution improvement is obtained consecutively for *N_max* iterations.

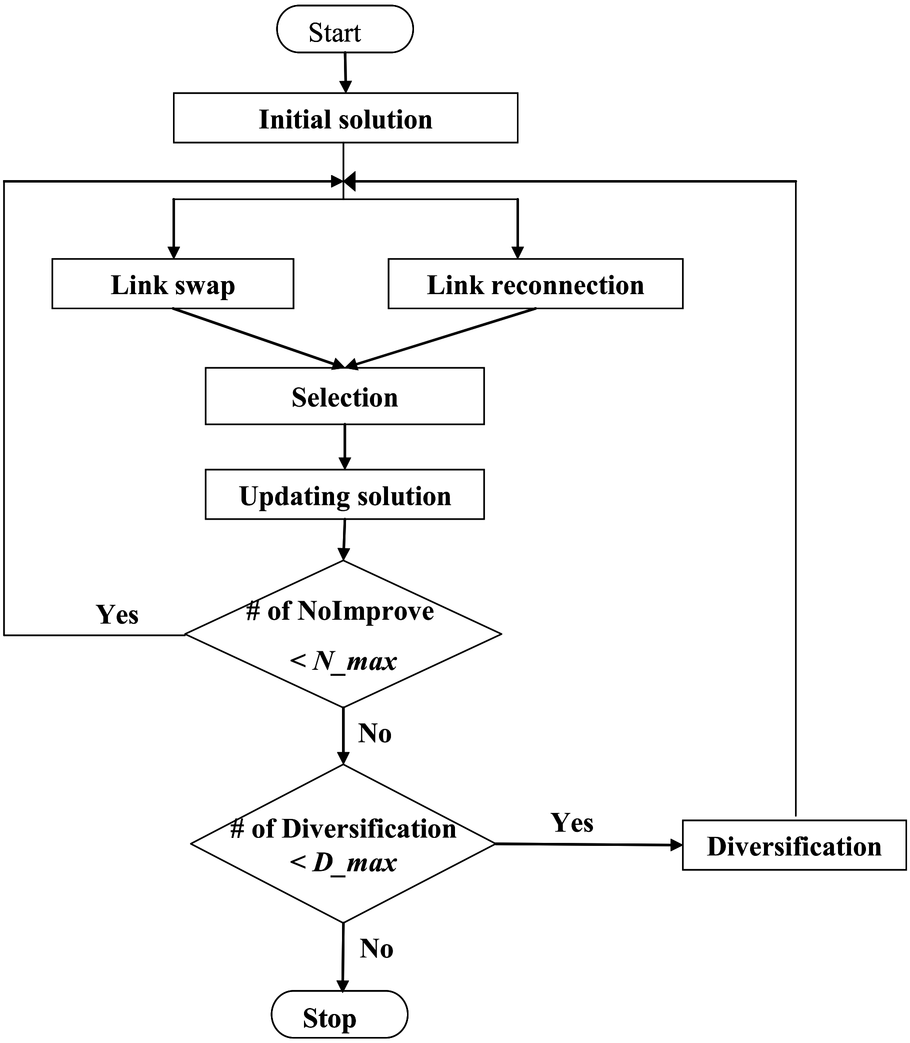


Figure 3. Proposed tabu search procedure

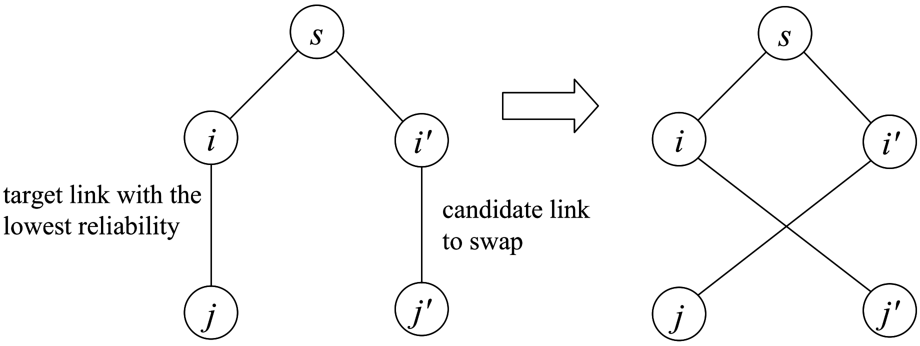


Figure 4. Link swap move

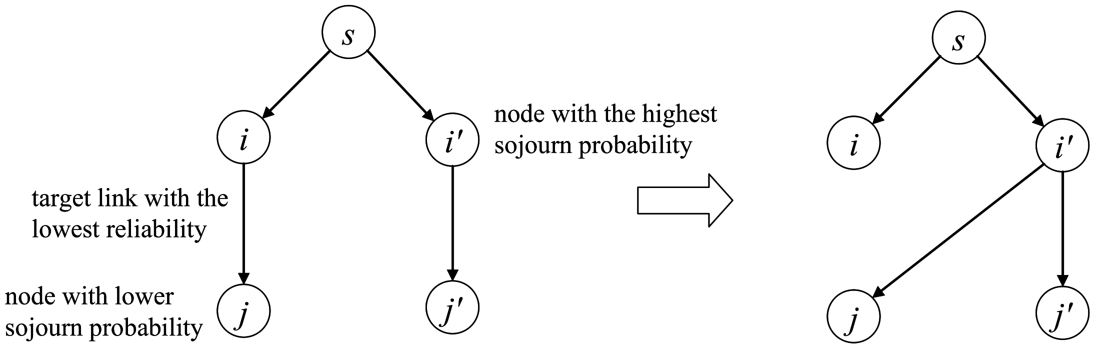

s
i
i′
j
j′
node with the highest sojourn probability
target link with the lowest reliability
node with lower sojourn probability
s
i
i′
j
j′

Figure 5. Link reconnection move

4.3 Diversification with Long-Term Memory

The purpose of diversification with a long-term memory is to drive the search into new solution regions escaping from local optimality. It is initiated when no solution improvement is obtained during *N_max* iterations of the intensification process. To restart the tabu search in new solution region, historical link frequency information is employed. The frequency count of a link represents how often the link is considered as a solution in previous pass of tabu search. Now, to restart the tabu search, it is promising to have an initial tree with links that are less frequently used as a solution. After ordering the frequency from the lowest to the highest, a feasible spanning tree is constructed by inserting the link with the lowest frequency first in each session. After the construction of feasible spanning trees, tabu search is continued with the intensification procedure. When the number of diversifications is equal to *D_max*, the procedure is terminated.

5. Computational Results

To test the proposed tabu search for reliable overlay multicast trees, overlay multicast networks are generated as in Table 2. Problems with 50, 100 and 200 multicast nodes in a session are considered with 1, 2, 3 and 4 sessions. For each case 10 problems are generated by randomly selecting node degree, link capacity and delay bound of each session as in the table. The widely spread video streaming codec MPEG-I [28] is assumed with service data rate 384 kbps or 600 kbps. Sojourn probability for each overlay multicast member is assumed, exponentially distributed with

$p_i^k = 1 - e^{-\lambda_i^k}$, where λ_i^k is failure rate of member i in session k. The memoryless property of exponential distribution well characterizes the reliability of an overlay multicast tree. With λ_i^k distributed over [0.5 ~ 5.0], sojourn probability p_i^k is given by [0.39 ~ 0.99]. Sojourn probability of the source node is assumed to be one. All procedures are run on a Pentium IV-1.8 GHz PC with 1024 Mbytes of memory under Windows XP.

We first test two initial solution strategies: Reliable Initial Tree and Random Initial Tree. Problems with 100 nodes are tested with different number of sessions. In each case, ten instances are experimented and average performance is shown in Figure 6. The figure shows that the solution by Reliable Initial Tree is better than that of Random Initial Tree. For rest of the experiment we thus apply Reliable Initial Tree for the initial overlay multicast trees.

Before applying tabu search, we need to optimize the tabu parameters: tabu list size, *N_max* for the intensification procedure and *D_max* for the diversification. Tabu list size represents the number of iterations during which a target node is forbidden to be adopted in move operation. By assuming that an appropriate tabu list size is proportional to the number of nodes in a session, we perform tests with five different values. Figure 7 shows that $0.2n_k$ is suitable for tabu list size with 100 nodes. Additional tests show that the following tabu list sizes are appropriate: $0.18n_k$ for problems with 50 nodes and $0.15n_k$ for 200 nodes.

Test for *N_max* is performed as in Figure 8. The figure shows that appropriate value for *N_max* is $0.3n_k$ for 50 nodes, $0.2n_k$ for 100 nodes and $0.15n_k$ for 200 nodes. Test results reveal that the number of sessions has little influence on the *N_max*. This seems to be mainly because only nodes with lower sojourn probabilities in all sessions are selected as target nodes.

The number of diversification is related to the solution quality in tabu search. Test of *D_max* is performed with one and four sessions as in Figure 9. Among ten problems, the portion that gives no further improvement for the successive diversification is plotted in the figure. Clearly, more diversification is required as the number of nodes increases. From the experiments it seems to be reasonable to apply *D_max* = 8 for 50 nodes, 12 for 100 nodes, and 14 for 200 nodes in case of one, two or three sessions. For problems with four sessions, *D_max* = 9, 13 and 15 are applied for 50, 100 and 200 nodes respectively.

Table 2. Parameters for overlay multicast networks

Number of multicast nodes in session k (n_k)	50, 100 and 200
Number of sessions (K)	1, 2, 3 and 4
Node degree (D_i)	3, 4 and 5
Link delay (d_{ij}^k)	1 hop
Delay bound (L_k)	$0.1n_k$, $0.15n_k$ and $0.2n_k$
Service data rate (r_k)	384 kbps, 600 kbps
Link capacity (C_{ij})	0.6 ~ 1.2 Mbps
Failure rate (λ_i^k)	0.5 ~ 5.0
Sojourn probability (p_i^k)	$1-e^{-\lambda_i^k}$

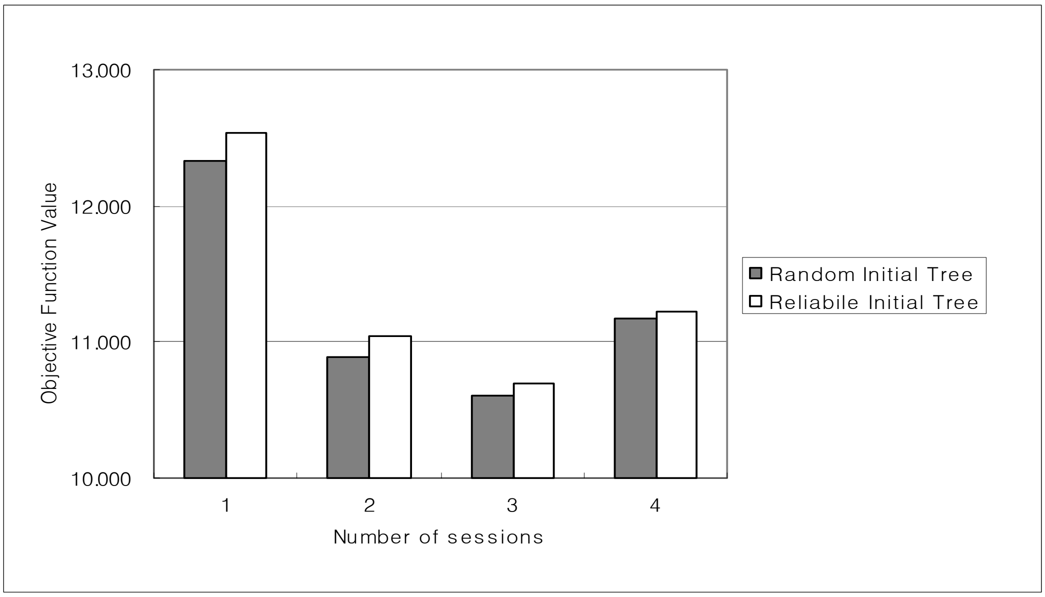


Figure 6. Test of initial solutions with 100 nodes

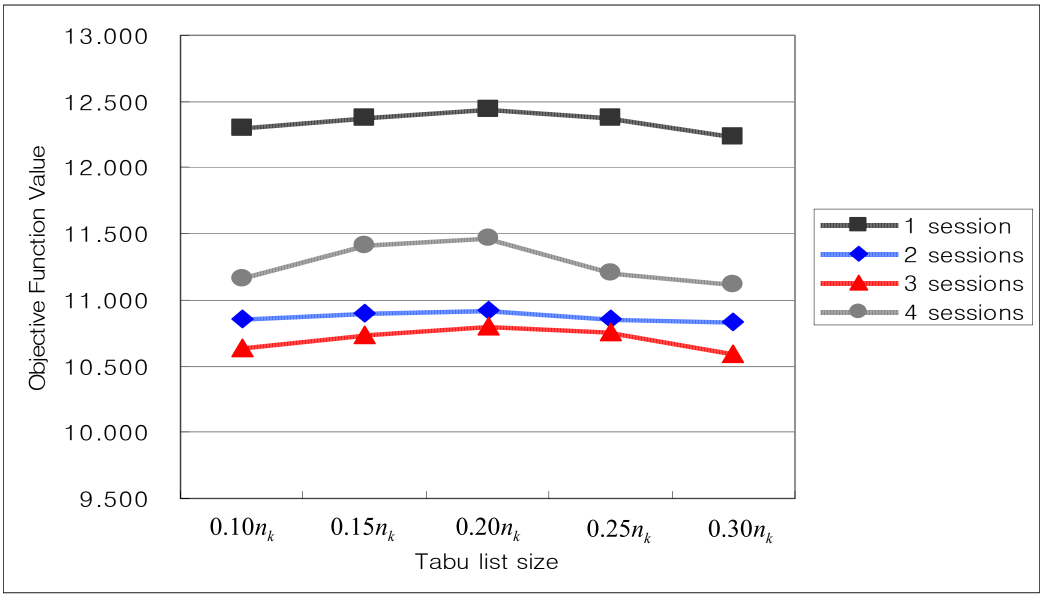


Figure 7. Test of tabu list size with 100 nodes

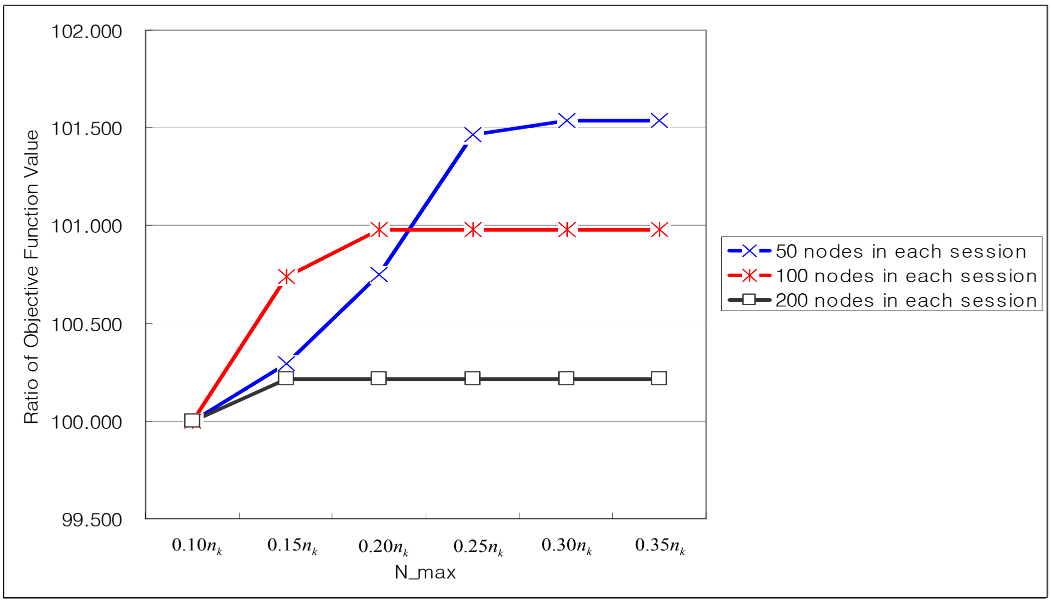
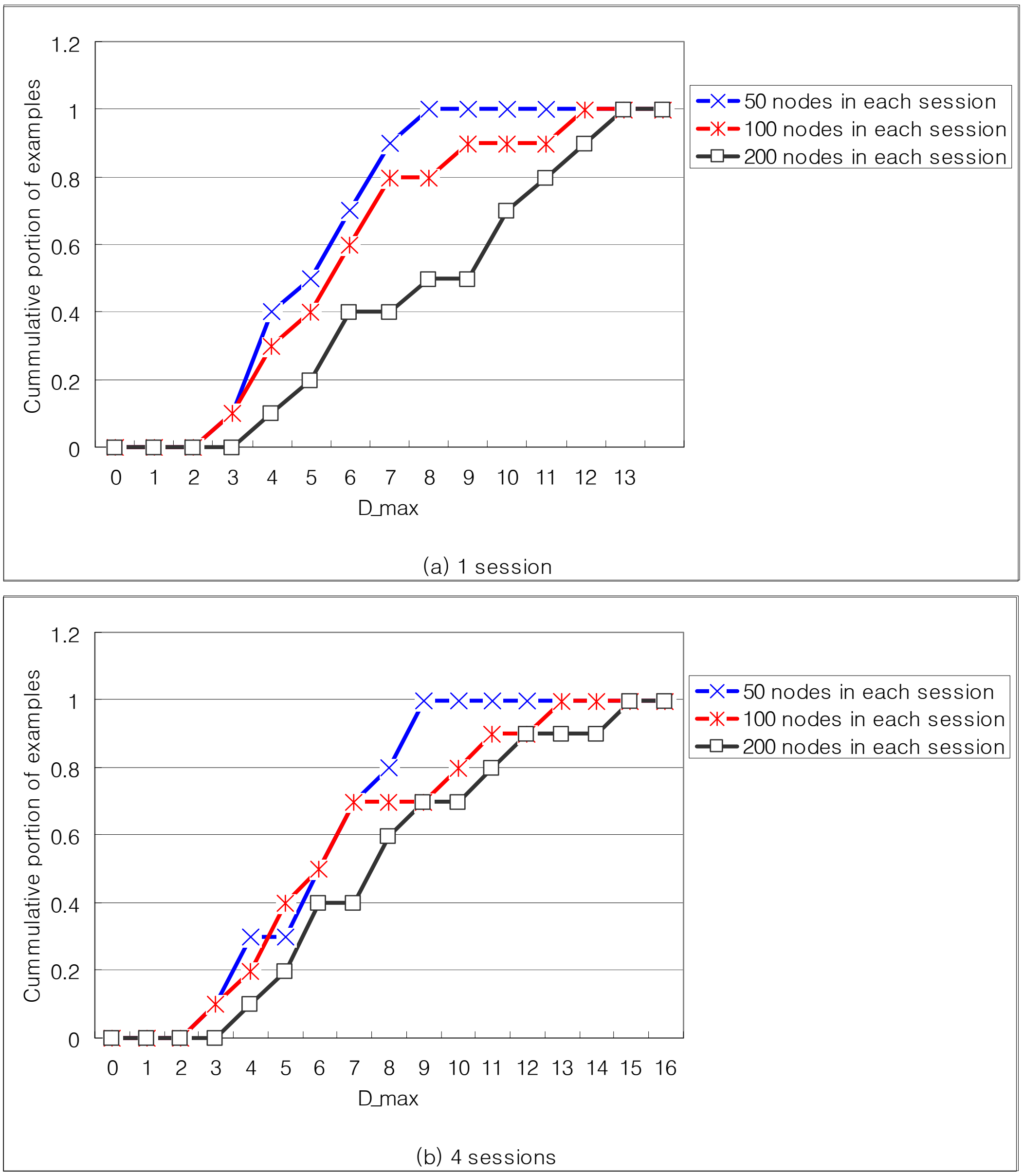


Figure 8. Test of *N_max*

Table 3, 4 and 5 respectively shows results of reliable overlay multicast trees with 50, 100 and 200 nodes in each session. CPLEX [29] is employed to compare the performance of the proposed tabu search. From the tables, it is clear that the performance of proposed tabu search is outstanding. The gap from the optimal solution or the lower bound by the CPLEX is within 1% except some problems of 50 and 200 nodes. However, the performance of CPLEX is restrictive. Due to the exponential growth of branches in the process of CPLEX, it fails to obtain the optimal solution in 10,000 seconds for problems with 100 nodes. Furthermore, in cases of 200 nodes with two or more sessions CPLEX fails to solve problems because of memory problem. The time efficiency of tabu search is also illustrated in the tables. The increase of solution time is linear to the number of multicast members and the number of sessions.



(a) 1 session

(b) 4 sessions

Figure 9. Test of *D_max*

Effectiveness of the diversification in the proposed tabu search is demonstrated in Figure 10 with 100 nodes. The figure shows results of tabu search with and without diversification. With diversification, the minimum link reliability measure of the overlay multicast tree is increased approximately by 2.5%.

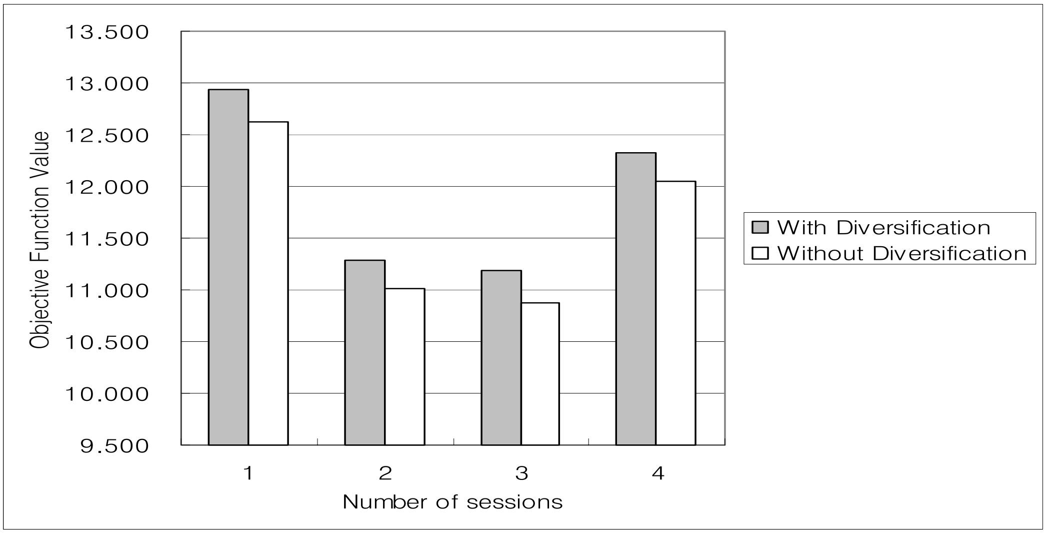


Figure 10. Effectiveness of Diversification

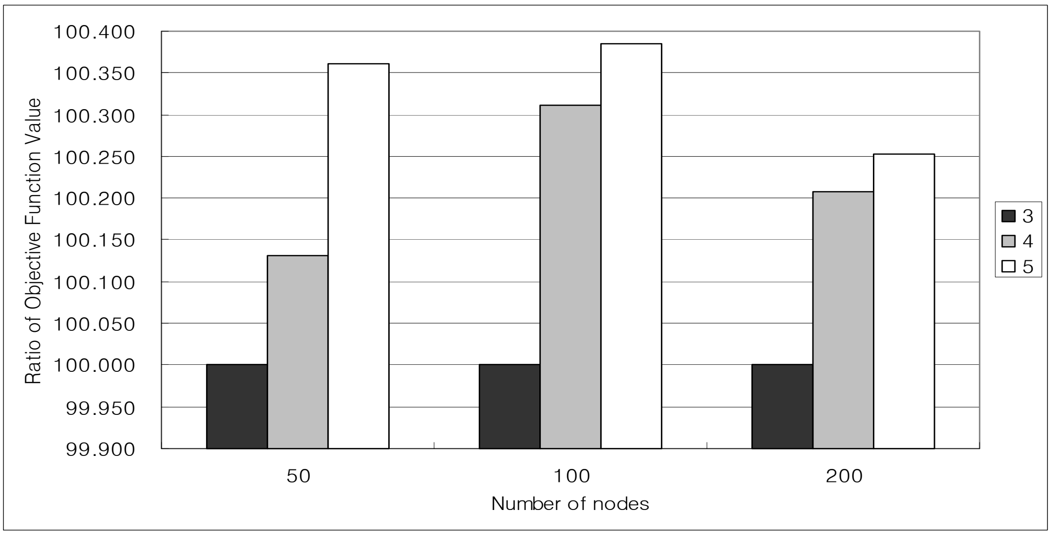


Figure 11. The effect of node degree

Table 3. Computational results of tabu search with 50 nodes in each session

Problem	1 session			2 sessions			3 sessions			4 sessions		
	Tabu Search (TS)	CPLEX	GAP**	Tabu Search (TS)	CPLEX	GAP**	Tabu Search (TS)	CPLEX	GAP**	Tabu Search (TS)	CPLEX	GAP**
1	12.623 (1.05)	12.706 (1127.45)	0.007	12.949 (1.61)	13.052 (1300.20)	0.008	10.581 (1.70)	10.646 (2406.36)	0.006	10.619 (2.09)	10.646 (8232.84)	0.003
2	10.713 (0.95)	10.897 (1136.13)	0.017	17.543 (1.52)	17.712 (1346.17)	0.010	10.080 (1.69)	10.128 (2369.03)	0.005	9.991 (2.08)	10.128 (8462.23)	0.014
3	13.956 (1.33)	14.072 (1454.70)	0.008	10.283 (1.63)	10.313 (1740.06)	0.003	10.019 (1.67)	10.078 (2135.01)	0.006	10.556 (2.06)	10.597 (10000.00*)	0.004
4	10.752 (1.20)	10.814 (1932.72)	0.006	10.522 (1.55)	10.572 (1869.05)	0.005	11.485 (1.83)	11.562 (2141.77)	0.007	9.980 (1.96)	10.048 (10000.00*)	0.007
5	9.961 (1.02)	10.081 (1661.77)	0.012	11.285 (1.58)	11.344 (3944.59)	0.005	14.269 (1.77)	14.335 (2286.23)	0.005	10.425 (1.97)	10.472 (7278.14)	0.005
6	10.967 (1.09)	11.080 (1778.69)	0.010	11.434 (1.66)	11.478 (3208.25)	0.004	10.865 (1.70)	10.968 (4520.80)	0.009	10.653 (2.05)	10.685 (7181.33)	0.003
7	16.040 (1.05)	16.314 (1797.50)	0.017	13.902 (1.69)	13.963 (1255.44)	0.004	10.356 (1.75)	10.412 (4539.44)	0.005	10.792 (2.08)	10.848 (7353.61)	0.005
8	10.259 (1.00)	10.438 (1657.70)	0.017	10.211 (1.59)	10.265 (1746.11)	0.005	10.681 (1.81)	10.754 (4517.42)	0.007	10.870 (2.07)	10.954 (7373.75)	0.008
9	10.425 (1.19)	10.572 (1520.56)	0.014	13.434 (1.56)	13.491 (2587.59)	0.004	10.332 (1.79)	10.377 (5573.94)	0.004	12.229 (2.02)	12.302 (7579.44)	0.006
10	11.023 (1.18)	11.210 (1819.69)	0.017	13.001 (1.58)	13.116 (1192.03)	0.009	10.636 (1.83)	10.712 (5587.48)	0.007	10.440 (2.08)	10.550 (1589.41)	0.011

*Terminated by the time limit

** GAP = (CPLEX – TS) / TS

The numbers in the parenthesis represent the CPU seconds

Table 4. Computational results of tabu search with 100 nodes in each session

Problem	1 session			2 sessions			3 sessions			4 sessions		
	Tabu Search (TS)	CPLEX	GAP**	Tabu Search (TS)	CPLEX	GAP**	Tabu Search (TS)	CPLEX	GAP**	Tabu Search (TS)	CPLEX	GAP**
1	10.491 (4.88)	10.516 (10000.00*)	0.002	11.391 (6.95)	11.455 (10000.00*)	0.006	11.615 (8.20)	11.690 (10000.00*)	0.006	12.030 (11.28)	12.146 (10000.00*)	0.010
2	12.088 (4.27)	12.151 (10000.00*)	0.005	13.054 (6.47)	13.058 (10000.00*)	0.000	10.875 (8.30)	10.887 (10000.00*)	0.001	11.338 (11.30)	11.340 (10000.00*)	0.000
3	10.658 (4.58)	10.735 (10000.00*)	0.007	11.159 (6.50)	11.216 (10000.00*)	0.005	11.299 (8.58)	11.315 (10000.00*)	0.001	12.442 (11.19)	12.464 (10000.00*)	0.002
4	16.231 (4.84)	16.325 (10000.00*)	0.006	9.936 (6.20)	9.945 (10000.00*)	0.001	10.940 (8.30)	10.996 (10000.00*)	0.005	11.894 (11.92)	11.903 (10000.00*)	0.001
5	15.184 (4.06)	15.302 (10000.00*)	0.008	10.422 (6.72)	10.430 (10000.00*)	0.001	10.433 (8.86)	10.448 (10000.00*)	0.001	12.633 (11.80)	12.652 (10000.00*)	0.002
6	10.933 (4.30)	11.005 (10000.00*)	0.007	10.104 (6.75)	10.153 (10000.00*)	0.005	11.368 (8.92)	11.444 (10000.00*)	0.007	9.722 (11.67)	9.736 (10000.00*)	0.001
7	12.803 (4.08)	12.896 (10000.00*)	0.007	10.014 (6.64)	10.046 (10000.00*)	0.003	10.511 (8.27)	10.518 (10000.00*)	0.001	10.808 (11.09)	10.829 (10000.00*)	0.002
8	11.915 (4.67)	11.935 (10000.00*)	0.002	13.599 (6.63)	13.679 (10000.00*)	0.006	10.624 (8.33)	10.690 (10000.00*)	0.006	10.733 (11.63)	10.782 (10000.00*)	0.005
9	9.898 (4.64)	9.938 (10000.00*)	0.004	12.517 (6.03)	12.551 (10000.00*)	0.003	11.873 (8.52)	11.892 (10000.00*)	0.002	13.748 (11.88)	13.793 (10000.00*)	0.003
10	18.105 (4.91)	18.179 (10000.00*)	0.004	10.552 (6.27)	10.566 (10000.00*)	0.001	10.256 (8.97)	10.295 (10000.00*)	0.004	10.050 (11.16)	10.138 (10000.00*)	0.009

*Terminated by the time limit

** GAP = (CPLEX – TS) / TS

The numbers in the parenthesis represent the CPU seconds

Table 5. Computational results of tabu search with 200 nodes in each session

Problem	1 session			2 sessions	3 sessions	4 sessions
	Tabu Search (TS)	CPLEX	GAP**	Tabu Search (TS)	Tabu Search (TS)	Tabu Search (TS)
1	15.375 (10.02)	15.381 (10000.00*)	0.000	12.011 (13.20)	10.999 (17.89)	13.434 (20.13)
2	14.161 (10.03)	14.171 (10000.00*)	0.001	12.151 (13.23)	9.749 (17.83)	12.467 (20.75)
3	10.745 (10.64)	11.132 (10000.00*)	0.036	9.950 (13.25)	15.098 (17.42)	12.306 (20.53)
4	11.734 (10.65)	11.742 (10000.00*)	0.001	9.788 (13.22)	11.504 (17.27)	10.148 (20.08)
5	11.070 (10.70)	11.074 (10000.00*)	0.000	10.409 (13.23)	10.711 (17.88)	11.260 (20.36)
6	18.402 (10.49)	18.442 (10000.00*)	0.002	13.125 (13.31)	10.004 (17.86)	9.693 (20.63)
7	11.014 (10.39)	11.036 (10000.00*)	0.002	9.663 (13.22)	14.281 (17.84)	12.214 (20.09)
8	17.931 (10.40)	17.991 (10000.00*)	0.003	9.945 (13.88)	10.160 (17.81)	10.061 (20.45)
9	17.430 (10.03)	17.431 (10000.00*)	0.000	14.299 (13.19)	10.831 (17.02)	10.102 (20.55)
10	16.305 (10.05)	16.332 (10000.00*)	0.002	9.710 (13.30)	9.686 (17.91)	10.210 (20.73)

*Terminated by the time limit

** GAP = (CPLEX – TS) / TS

The numbers in the parenthesis represent the CPU seconds

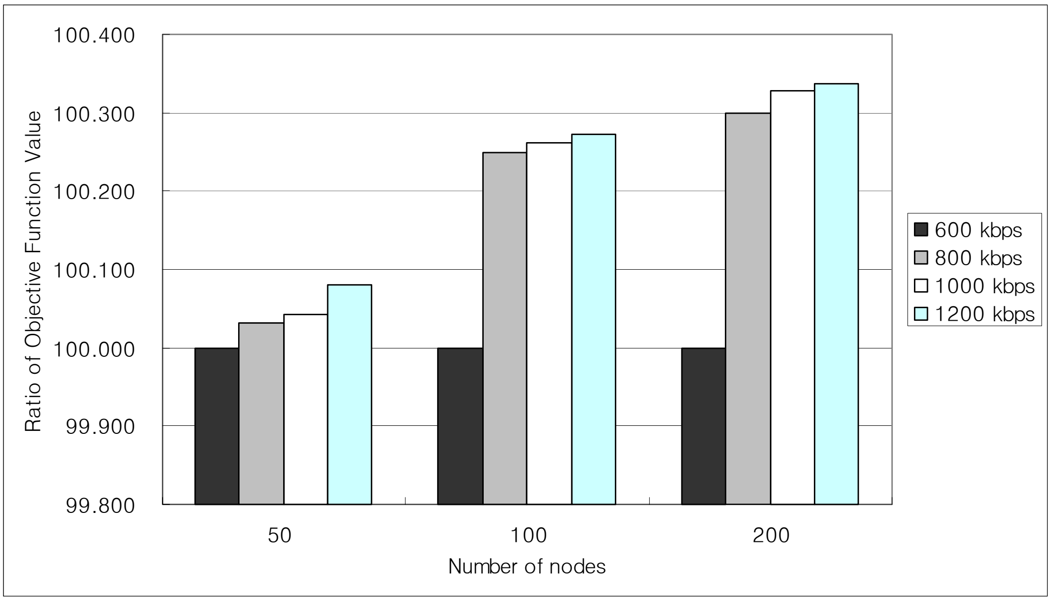


Figure 12. The effect of link capacity

Now, we consider sensitivity of the three constraints of the overlay multicast tree problem, i.e., node degree, link capacity and delay bound in the formulation of Section 3. Figure 11, 12 and 13 show the effect of node degree, link capacity and delay bound, respectively. To analyze the effect of node degree, the degree of all nodes is fixed to 3, 4 and 5 respectively except for the source node. The delay bound is fixed to $0.15n_k$. As shown in Figure 11, the increase of link reliability seems to be sensitive to the node degree in problems with 50 and 100 nodes. The increase is most sensitive in problems with 100 nodes when the node degree is changed from three to four. The effect of link capacity is shown in Figure 12. The delay bound is also fixed to $0.15n_k$. The figure shows that the increase of minimum link reliability is critical when the link capacity is increased from 600 kbps to 800 kbps. Finally, Figure 13 shows sensitivity of the objective function to the delay bound. Three different delay bounds $0.1n_k$, $0.15n_k$ and $0.2n_k$ are experimented. From the figure, it is clear that the link reliability is more sensitive to the delay bound as the number of multicast members increases.

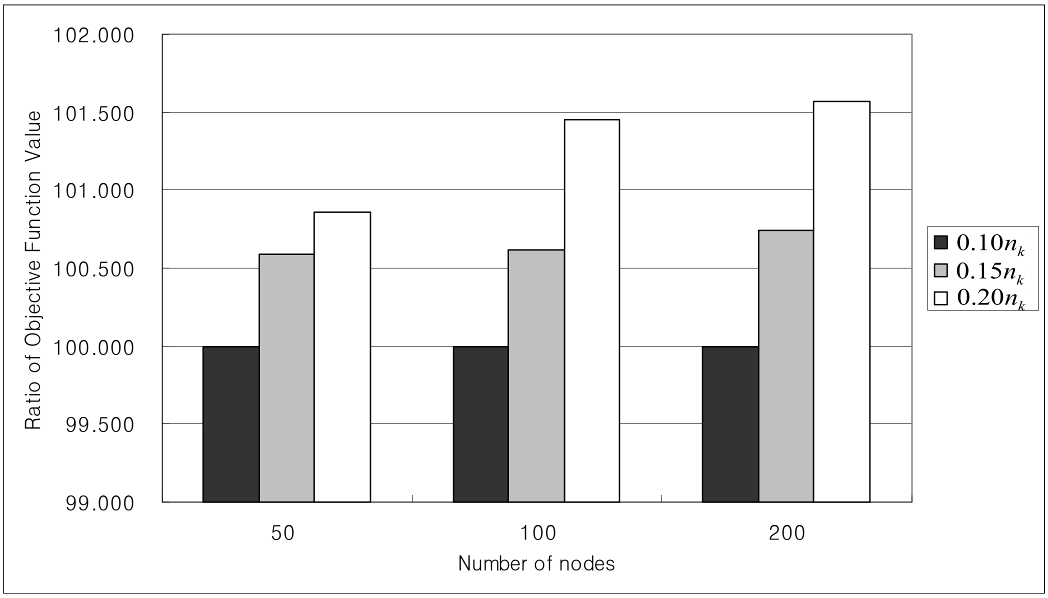


Figure 13. The effect of delay bound

6. Conclusion

The problem of maximizing the minimum link reliability in overlay multicast tree for private Internet broadcasting is examined. To increase network connectivity among users of Internet broadcasting, overlay multicast transmission technology is introduced. The application layer overlay multicast technology has bandwidth efficiency and scalability compared to the traditional IP multicast. However, to support dynamic joining or leaving of group members of each session of Internet broadcasting session, we need to rearrange the overlay multicast trees such that they have high link connectivity. We assume that each user node has a sojourn probability that resides in its multicast group during a fixed period of time. From this sojourn probability, a link reliability measure based on an utility function is obtained.

The problem of maximizing the minimum link reliability is formulated as a binary integer programming. Three important constraints of overlay multicast network are considered: node degree, link capacity and delay bound. A tabu search heuristic based on repeated intensification and diversification is proposed. Link swap and link reconnection moves are considered by employing a short-term memory of the tabu list. Diversification based on historical long-term memory is also implemented to investigate new and better solution that maximizes the minimum link reliability.

Computational experiments of the proposed tabu search are performed for overlay multicast networks with 50, 100 and 200 nodes in each session. An outstanding performance is illustrated by the proposed tabu search. The gap from the optimal solution or the available lower bound is within 1% except some problems of 50 and 200 nodes. The time efficiency of the proposed tabu search is also verified. The increase of solution time is linear to the number of multicast members and sessions.

References

[1] Dapeng Wu, Yiwei Thomas Hou, Wenwu Zhu, Ya-Qin Zhang and Jon M. Peha, “Streaming Video over the Internet: Approaches and Directions”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 11, No. 3, 2001, pp. 282-300.

[2] Sangmoon Lee, Sinjun Kang, Byungseok Min and Hagbae Kim, “Development of s Scalable

Clustering Streaming Server for the Internet Personal-Live Broadcasting", In Proceedings of IEEE 12th International Symposium on Software Reliability Engineering, Hong Kong, Nov., 2001, pp. 33-34.

[3] SpotLife, http://www.spotlife.com.

[4] Personal Internet Broadcasting, http://www.ibaglobal.com/pib.html.

[5] SHOUTcast, http://www.shoutcast.com.

[6] Yang-hua Chu, Aditya Ganjam, T. S. Eugene Ng, Sanjay G. Rao, Kunwadee Sripanidkulchai, Jibin Zhan and Hui Zhang, "Early Experience with an Internet Broadcast System Based on Overlay Multicast", Technical Report CMU-CS-03-214, Carnegie Mellon University, Dec., 2003.

[7] Christophe Diot, Brian Neil Levine, Bryan Lyles, Hassan Kassem and Doug Balensiefen, "Deployment Issues for the IP Multicast Service and Architecture", IEEE Network, Vol. 14, No. 1, 2000, pp. 78-88.

[8] Carlos A.S. Oliveira and Panos M. Pardalos, "Algorithms for the Streaming Cache Placement Problem on Multicast Networks", In Proceedings of Seventh INFORMS Telecommunications Conference (ITC'04), Boca Raton, Florida, Mar., 7-10, 2004.

[9] A. El-Sayed, V. Roca and L. Mathy, "A Survey of Proposals for an Alternative Group Communication Service", IEEE Network, Vol. 17, No. 1, Jan., 2003, pp. 46-51.

[10] Dimitrios Pendarakis, Sherlia Shi, Dinesh Verma and Marcel Waldvogel, "ALMI: An Application Level Multicast Infrastructure", In Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems (USITS), 2001.

[11] V. Roca and A. El-sayed, "A Host-based Multicast (HBM) Solution for Group Communication", In Proceedings of IEEE International Conference on Networking, 2001.

[12] Paul Francis, Yuri Pryadkin, Pavlin Radoslavov, Ramesh Govindan and Bob Lindell, "Yoid: Your Own Internet Distribution", http://www.isi.edu/div7/yoid/docs/paper.ps.

[13] John Jannotti, David K. Gifford, Kirk L. Johnson, M. Frans Kaashoek and James W. O'Toole, "Overcast: Reliable Multicasting with an Overlay Network", In Proceedings of 5th Symposium on Opening System Design and Implementation, 2000.

[14] Suman Banerjee, Bobby Bhattacharjee and Christopher Kommareddy, “Scalable Application Layer Multicast”, In proceedings of ACM SIGCOMM, 2002, pp. 205-220.

[15] Yang-hua Chu, Sanjay G. Rao, Srinivasan and Hui Zhang, “A Case for End System Multicast”, IEEE Journal on Selected Areas in Communications, Vol. 20, No. 8, Oct., 2002, pp. 1456-1471.

[16] Yatin Chawathe, Steven McCanne and Eric Brewer, “An Architecture for Internet Content Distribution as an Infrastructure Service”, unpublished paper, 2000.

[17] Lawrence A. Rowe, Diane Harley, Peter Pletcher and Shannon Lawrence, “BIBS: A Lecture Webcasting System”, Technical Report, Berkeley Multimedia Research Center (BMRC), University of California, Berkeley, June, 2001.

[18] Akamai, http://www.akamai.com.

[19] Real broadcast network, http://www.real.com/

[20] Common Log Format, http://www.w3.org/Daemon/User/Config/Logging.html.

[21] Extended Common Log Format, http://www.w3.org/TR/WD-logfile.html.

[22] Suman Banerjee, Christopher Kommareddy, Koushik Kar, Bobby Bhattacharjee and Samir Khuller, “Construction of an Efficient Overlay Multicast Infrastructure for Real-time Applications”, In Proceedings of IEEE INFOCOM, 2003, pp. 1521-1531.

[23] Carlos A.S. Oliveira and Panos M. Pardalos, “A Survey of Combinatorial Optimization Problems in Multicast Routing”, Computers and Operations Research, Vol. 32, No. 8, 2005, pp. 1953-1981.

[24] Mokhtar S. Bazaraa, John J. Jarvis and Hanif D. Sherali, “Linear Programming and Network Flows”, 2nd edition, John Wiley & Sons, 1990.

[25] Michael R. Garey and David S. Johnson, “Computers and Intractability: A Guide to the Theory of NP-Completeness”, W. H. Freeman and Company, 1979.

[26] Fred Glover and Manuel Laguna, “Tabu Search”, Kluwer Academic Publishers, 1997.

[27] Fred Glover, “Tabu Search: A Tutorial”, Interfaces, 1990.

[28] Digital Encoding Services, Inc, http://www.digital-encoding.com/streaming_enc.htm.

[29] CPLEX 8.1, http://www.ilog.com/products/cplex.