Mathematical Problems of Computer Science 28, 2007, 65 – 74.

Network Resource Optimization for Quality of Service in Multimedia Multicasting

Vahe Aghazarian* and Hossein Pedram**

* Islamic Azad University, Sciences and Researches Branch
e-mail: v_aghazian@iauctb.ac.ir
** Amir Kabir University of Technology
e-mail: pedram@ce.aute.ac.ir

Abstract

Multicast is a bandwidth efficient mechanism for delivering the same data to multiple receivers simultaneously. In this paper, a well-organized method to build source specific multicast trees is proposed. This algorithm aims at reaching a high traffic balance in the network in order to avoid bandwidth bottlenecks and consequent network partitions, one of the main causes for low network performance. To do so, it computes multicast trees by dynamically selecting the least loaded available paths, obtaining an optimal distribution of network resources. Strictly integrated with the DiffServ Quality of Service (QoS) approach, the proposed multirate native multicast algorithm maps the QoS service requested by receivers into the proper DiffServ class, so as to respect the expected QoS requirements. The results are a better leverage of the network bandwidth resources, an improved QoS perceived by multicast group members, and time and resource saving due to its low computational complexity, as shown through general C++ based simulation.

1. Introduction and Background

In unicast transmissions the sender transmits data to a single receiver and, if multiple receivers want to receive the same data content, the sender has to transmit multiple copies of data. In multicast transmission, differently, the sender transmits only one copy of data that is delivered to multiple receivers. One of the most challenging objectives in multicasting is to minimize the amount of network resources employed to compute and setup multicast trees [1][2]. In multicast communication the routing problem is to find the minimum-weight tree that spans all the nodes in the multicast group [3][4].

In source specific multicast communications only one node in the multicast group sends data while all the other member nodes receive data. A tree that spans all member nodes is said multicast tree. A source-rooted tree has the source node as root and is optimized for source specific multicast communications.

The classical optimization problem in multicast routing is the Steiner tree problem in networks (SPN) [5] whose objective is to find the least-cost tree connecting the source with the group of destinations with the minimum total cost over all links. If each destination has a bandwidth requirement, then the problem is to find the least-cost tree that respects the bandwidth requirements on each path from the source to the receiver. Since both these two problems are

NP-complete, efficient algorithms to solve these problems in polynomial time attain only approximate solution [4][5].

In source specific communications, multicast sessions may have a large number of receivers with heterogeneous reception capacities. To accommodate this heterogeneity, a layering scheme can be used [6][7]. In a layering scheme, data transmission through the network takes place over logical channels. A sender can simultaneously transmits data on multiple channels. Each channel has its own transmission rate, usually computed at the sender side. Receivers subscribe to the layers cumulatively, receiving data at a cumulative rate.

The proposed multicast algorithm builds source-rooted multicast trees for source specific applications. This algorithm takes into account the network link available bandwidth in order to avoid bandwidth bottlenecks in the network. The idea is to keep low the average link traffic utilization by fairly distributing data flows among those least loaded links. Our proposed algorithm is integrated with the DiffServ Quality of Service (QoS) approach [8][9]. The QoS service requested by receivers, mapped into the proper DiffServ class, is taken into account in the multicast tree computation. The presented algorithm allows members with less stringent QoS requirements to reuse resources already exploited by members with more stringent QoS requirements. Simulation results show a better leverage of the network bandwidth resources, an improved QoS perceived by multicast group members with respect to the performance of existing multicast algorithms, and a resource saving due to low computational complexity, which dramatically characterizes the algorithm.

The main innovative features in this work are:

- Seamlessly integration of DiffServ Quality of Service approach in the proposed multicast algorithms;
- Leverage of the network bandwidth resources and an improved QoS perceived by multicast group members;
- Low computational complexity which effectively leads to time and resource saving.

The remainder of the paper is organized as follows.

In section II, the proposed multicast algorithm for source specific applications in both non-QoS-aware (II-A) and QoS-aware (II-B) networks is presented. In section III we show numerical results through a general simulation ran on an ad-hoc C++ simulator. Finally, in section IV we conclude the paper.

2. Proposed Algorithm

This algorithm focuses on the problem of bandwidth bottlenecks, one of the main reasons for low network performance. The multicast source rooted tree is built by choosing those paths that result least loaded to obtain a smooth load balancing in order to achieve an optimal distribution of network resources and thus maximize the number of constructed multicast trees.

A. Non-QoS-aware Networks Scenario

Let us consider a multirate multicast scenario where receivers ask for different rates [6][7]. If K channels with rate $\{w_1,...w_K\}$ are used in the layering scheme, then K cumulative rate $L_1,...,L_K$ are available.

The algorithm proceeds as it follows.

Let s be the source node and let us suppose that the multicast group is made up of M receivers.

* Step 0:

Receivers are ordered from the highest rate to the lowest one. At the end of this step we have an ordered set of M receivers $\{r_1,...,r_M\}$, with demanded cumulative rates $F_1 \geq F_2 \geq ... \geq F_M$

$((F_j\ \{L_1,...,L_K\},\ j = 1,...,M)$. Receivers from r_1 to r_M are connected to the source node progressively. In this way, receiver r_i can reuse resources already exploited on paths from the source to receivers $r_1,...,r_{i-1}$. These receivers, in fact, ask for rates $F_1 \geq F_2 \geq ... \geq F_{i-1} \geq F_i$. Let $S_{path}(s,r_k)$ be the set of all feasible paths from s to r_k. A path $p(s,r_k)$ from source s to receiver r_k is feasible if $b_{uv} \geq F_K$ for all its links, where b_{uv} is the available bandwidth in link (u,v).

*Step k, $k = 1,...,M$:

The algorithm chooses the path $\overline{p(s,r_k)} \in S_{path}(s,r_k)$ that minimizes the following function:

$$f\{p(s,r_k)\} = \sum_{\{(u,v)\in p(s,r_k)\}} \frac{a_{uv}}{(1-\rho_{uv})^{\alpha}}$$
$$p(s,r_k) \in S_{path}(s,r_k) \qquad (1)$$

1) is the link (u,v) utilization and it is defined as:

$$\rho_{uv} = \frac{B_{uv}-(b_{uv}-F_k)}{B_{uv}} = \frac{B_{uv}-\beta_{uv}}{B_{uv}} \qquad (2)$$

Where B_{uv} is the total bandwidth capacity of link (u,v), F_k is the bandwidth exploited on link (u,v) by receiver r_k and $\beta_{uv} = b_{uv} - F_k$ is the residual bandwidth of link (u,v).

2) The exponent $\alpha = \alpha(|V|,|E|,\overline{F},\overline{B})$ in eq. 1 is a function of the number of nodes $|V|$, the number of links $|E|$, the average rate $\overline{F}$ demanded by receivers and the average network link bandwidth $\overline{B}$. According to accurate empirical observations (described hereafter) supported by general simulation results, the exponent α in eq. 1 has been chosen as:

$$\alpha = \alpha(|V|,|E|,\overline{F},\overline{B}) = a.\exp\left\{-b.\frac{|E|}{|V|.(|V|-a)}\right\}.\exp\left\{-c.\frac{\overline{F}}{\overline{B}}\right\} \qquad (3)$$

Coefficients a, b, c have been determined by solving the following minimization problem of the quadratic error between the exponent model in eq. 3 and the experimental results:

$$\min \sum_{i=1}^{N}\left[\alpha_i - \alpha(|V|_i,|E|_i,\overline{F}_i,\overline{B}_i\right]^2 \qquad a\in[0,3], b\in[0,+\infty] \qquad (4)$$

Starting from a set of points $\Gamma = \{(|V|_i,|E|_i,\overline{F}_i,\overline{B}_i), i = 1,...,N\}$, the optimum exponent was experimentally obtained for each point $(|V|_i,|E|_i,\overline{F}_i,\overline{B}_i) \in \Gamma$ of the set.

The coefficient α has been chosen belonging to the interval [0,3] after an accurate tuning of the model. Optimal values of a, b, c of the minimization problem (4) are $a = 3$, $b = 3.9$ and $c = 16.9$.

Let us now come back to eq. 3 to justify why the variable aggregations $\overline{F}/\overline{B}$ and $|E|/(|V|(|V|-1))$ are suitable to model the exponent in eq. 1.

By decreasing the value of α, the length of paths found by the algorithm is reduced because the difference between the metric of loaded links and the metric of unloaded links is reduced. It could be necessary to reduce the length of paths used by the multicast tree when the average rate $\overline{F}$ demanded by the receivers grows or when the average network bandwidth $\overline{B}$ decreases. In fact, the higher the value of $\overline{F}$, or the lower the value of $\overline{B}$, the more resources on each link are consumed by a single path. Thus, short paths have to be preferred. Simulations show that at the growing of the ratio $\overline{F}/\overline{B}$ the value of the optimal exponent α decreases exponentially. As far as concern the number of nodes $|V|$ and links $|E|$, if $|E|$ decreases, less paths are available and the resource utilization becomes a more sensitive issue. By increasing the value of the exponent α, the difference between the metric of loaded links and unloaded links increases. If the number of nodes $|V|$ grows but the number of links $|E|$ remains the same, the exponent increases because the number of paths decreases. Simulations show that at the decreasing of the ratio $|E|/(|V|(|V|-1))$ the value of the optimal exponent α grows exponentially.

3) The binary variable a_{uv} is equal to zero if link (u,v) already belongs to the tree, otherwise is 1. If the set $S_{path}(s,r_k)$ is not empty, let $\overline{p(s,r_k)}$ be the path from the source s to the receiver r_k that minimizes the function in eq. 1. On each link (u,v) belonging to $\overline{p(s,r_k)}$, if $a_{uv}=0$, the value of the available bandwidth is not updated and uses those bandwidth resources already exploited by a path $\overline{p(s,r_k)}$, with $F_j \geq F_k$. Conversely, if $a_{uv}=0$, then new resources must be consumed and the new value of the available bandwidth is $b'_{uv} = b_{uv} - F_k$. Then the binary variable a_{uv} is set to zero in order not to consider the cost of link (u,v) for those future paths which will exploit it.

Let us point out that at Step k, $k=1,...M$, the optimal path $\overline{p(s,r_k)} \in S_{path}(s,r_k)$ can be found using a shortest path algorithm (as Dijkstra or Bellman-Ford algorithm), where the length of each link in the network is set to:

$$d_{uv} = \begin{cases} \dfrac{a_{uv}}{(1-\rho_{uv})^{\alpha}} & \text{if } b_{uv} \geq F_k \\ \infty & \text{if } b_{uv} < F_k \text{ or } (u,v) \notin E \end{cases} \quad (5)$$

Our algorithm uses the Bellman-Ford algorithm, which finds a spanning tree of the shortest paths from the source node s to all other nodes of the graph [10]. The path $p(s,n)$ from node s to node n, solution of eq. 1, is the one that minimizes the function:

$$\sum_{\{(u,v)\in p(s,n)\}} d_{uv} \quad (6)$$

Computational complexity: if the number of receivers is M, our algorithm builds the multicast tree by computing for as many as M times the spanning tree, using the Bellman-Ford algorithm. Since the time complexity of the Bellman-Ford algorithm is $O(|V|.|E|)$, then the computational complexity of the algorithm is $O(M.|V|.|E|)$.

B. QoS-aware Networks Scenario

In this section the algorithm is integrated in a DiffServ network. Let us consider a multirate multicast scenario where receivers ask for the same data content but different rates and different Qualities of Service (QoS), mapped into the respective service class according to [8][9]. In particular, the algorithm can build source-rooted tree for both real time multicast communications, by adopting the Expedited Forwarding (EF) DiffServ service class, and non real time communications, by adopting the Assured Forwarding (AS) DiffServ service class. Our algorithm, when integrated in a DiffServ network, allows a receiver to reuse the bandwidth already exploited by receivers asking for higher service classes without any extra cost for the network. By building a path from the source s to receiver r, the algorithm can reuse some sub-paths already exploited by higher service class receivers. Thus receiver r obtains a better QoS and the network saves resources because bandwidth already exploited by other receivers is reused for receiver r.

The algorithm in a DiffServ network proceeds as it follows.

* Step 0:

Receivers are ordered according to their service class and rate. Let us consider a set R made up of M receivers. Let $CL=\{cl_1,...cl_L\}$ be the set of service classes demanded by receivers, where cl_1 is the highest service class and cl_L is the lowest one. CL is a subset of all service classes supported by DiffServ architecture [8][9].
The set of receivers R is partitioned into $L=|CL|$ subsets:

$$R=\bigcup_{i=1}^{L} R_i \qquad (7)$$

Subset R_i is made up of those receivers asking for service class cl_i , $i = 1,...,L$. Receivers in each subset R_i, are ordered from the highest rate to the lowest one. Let us consider a partitioned and ordered set R made up of M receivers. Let r_k^i be the receiver k in the subset R_i with demanded rate F_k^i.$|R_i|$ will stand for the cardinality of R_i, and i will represent the associated service class.

* Step j, $j=1,...,L$:

The algorithm connects the source s with all those receivers asking for service class cl_i. At the beginning of these steps, the binary variable a_{uv} are set to 1 for each network link (u,v). For receiver r_k^i, $k = 1,...,|R_i|$, a_{uv} are set to zero for all those links (u,v) already used by receivers that ask for higher service class and higher or equal rate than r_k^i. In fact, resources already exploited by the tree on those links could be used by r_k^i without any extra cost.

Let $b_{uv}(cl_i)$ be the available bandwidth of link (u,v) for the service class cl_i. A path $p(s,r_k^i)$ from source s to a receiver r_k^i asking for cumulative rate F_k^i is feasible if $b_{uv}(cl_i)\geq F_k^i$ for all its links. The algorithm chooses that feasible path $\overline{p(s,r_k^i)}$ from s to r_k^i that minimizes the function:

$$f\{p(s,r_k^i)\}=\sum_{\{(u,v)\in p(s,r_k^i)\}}\frac{a_{uv}}{(1-\rho_{uv}(cl_i))^{\alpha}}\ ,\ p(s,r_k^i)\in S_{path}(s,r_k^i) \qquad (8)$$

where $S_{path}(s,r_k^i)$ is the set of all feasible paths from s to r_k^i. Exponent α is defined as in eq. 3 and the utilization $\rho_{uv}(cl_i)$ of the link (u,v) is calculated as follows:

$$\rho_{uv}=\frac{B_{uv}(cl_i)-[b_{uv}(cl_i)-F_k^i]}{B_{uv}(cl_i)} \qquad (9)$$

where $B_{uv}(cl_i)$ is the total bandwidth capacity of link (u,v) for service class cl_i and F_k^i is the cumulative rate of r_k^i.

For each link (u,v) belonging to path $\overline{p(s,r_k^i)}$, if $a_{uv}=0$ then the available bandwidth $b_{uv}(cl_i)$ is not updated and the new path uses those resources already exploited by other receivers. On the contrary, if $a_{uv}\neq 0$, then new resources have to be exploited and the new value of the available bandwidth becomes $b'_{uv}(cl_i)=b_{uv}(cl_i)-F_k^i$. Then the binary variable a_{uv} is set to zero.

Our proposed algorithm determines the path $\overline{p(s,r_k^i)}$ by using a modified shortest-path algorithm. Common shortest path algorithms could choose one of those possible paths with zero cost from s to n, but $\overline{p(s,r_k^i)}$ might not be correct. To determine a correct path the algorithm proceeds as it follows.

Starting from node s, the first link $(\widetilde{u},\widetilde{v})$ belonging to $\overline{p(s,r_k^i)}$ with $a_{uv}\neq 0$ is found. Among all those paths in the tree exploiting a bandwidth equal or greater than F_k^i and passing through

node $\tilde{u}$, it is chosen the one which uses the highest service class. Let us indicate this path with $\tilde{p}_i$. The new path
$\overline{\overline{p}(s,r_k^i)}$ from s to r_k^i is the concatenation of the sub-path from s to node $\tilde{u}$, belonging to path $\tilde{p}_i$, and the sub-path from node $\tilde{u}$ to r_k^i, belonging to $\overline{\overline{p}(s,r_k^i)}$.

3. Simulation Results

A. Random Network Model

To ensure a fairly evaluation of different routing algorithms, a random network has been generated to the Waxman's model [11][12].

In the Waxman's model network, nodes are randomly distributed across a Cartesian coordinate grid. Links are added to the graph by considering all possible pairs (u,v) of nodes and by using the probability function:

$$P_e(u,v) = \beta.\exp(-\frac{d_{uv}}{\alpha.L}) \qquad (10)$$

where $P_e(u,v)$ is the existence probability of a link between nodes u and v, d_{uv} is the Euclidean distance between node u and the node v, L is the maximum possible distance between a pair of nodes, α and β are parameters belonging to the range (0,1]. A high α value increases the number of connections to nodes further away, while a high β value increases the node degree.

Unlike the original Waxman's model, we suppose that each link added to the network is bi-directional. The bandwidth capacity B_{uv} of each link (u,v) is randomly generated using an uniform distributioan with mean bandwidth $\overline{B}$. The link (u,v) bandwidth capacity B_{uv} might be different from the link (v,u) bandwidth capacity B_{vu}.

In this work, two different one hundred node random networks are generated. Network 1 uses α=0.2 and β=0.4 while network 2 uses α=0.3 and β=0.6. Network 2 has a higher number of links than Network 1, according to the probability function (10). The average bandwidth in both networks is $\overline{B} = 100Mbps$.

B. Proposed Algorithm Performance Evaluation

In this section our algorithm is compared to the optimal solution of the Steiner tree problem when all link costs are equal to one. If every links have a cost equal to one, then the minimum-weight tree that spans all group members is the multicast tree that uses the least amount of network resources. The minimum-weight tree is found by solving the flow formulation of the Steiner tree problem proposed by Claus & Maculan [13] and Wong [3]. We further implemented the ILP problem in AMPL [14] and solved it with the CPLEX [15] solver.

Starting from a completely unloaded Waxman network [11], it is requested to build a fixed number of source-rooted trees, the Number of Requested Trees. Multicast groups are sequentially randomly generated. Multicast group members (source and receivers) are randomly chosen among network nodes. The number of receivers for each multicast group is uniformly distributed from 5 to 15 and the bandwidth request of each receiver is uniformly distributed from 0.1 to 2 Mbps. For each simulation several runs have been done to ensure a sufficient small trust region and short 95% confidence intervals.

To evaluate the performance of our algorithm and the optimal solution of the Steiner tree problem with unitary costs in the different simulated scenarios, we will use two metrics, the Rejection Rate and Network Load.

1) The Rejection Rate is defined as:

$$RejectionRate = \frac{Number\ of\ Rejected\ Trees}{Number\ of\ Requested\ Trees} \quad (11)$$

where Number of Requested Trees is the total number of source-rooted trees sequentially generated, while the Number of Rejected Trees is the number of requested source-rooted trees that cannot be built because there are nót enough resources in the network.

2) The Network Load $\overline{\rho}$ is defined as:

$$\overline{\rho} = \frac{\sum_{\{(u,v)\in E\}} \rho_{uv}}{|E|} \quad (12)$$

where E is the set of network links, $|E|$ its cardinality and ρ_{uv} is the Link Load of link (u,v), defined as:

$$\rho_{uv} = \frac{Used\ Bandwidth\ of\ link\ (u,v)}{Bandwidth\ Capacity\ of\ link\ (u,v)} \quad (13)$$

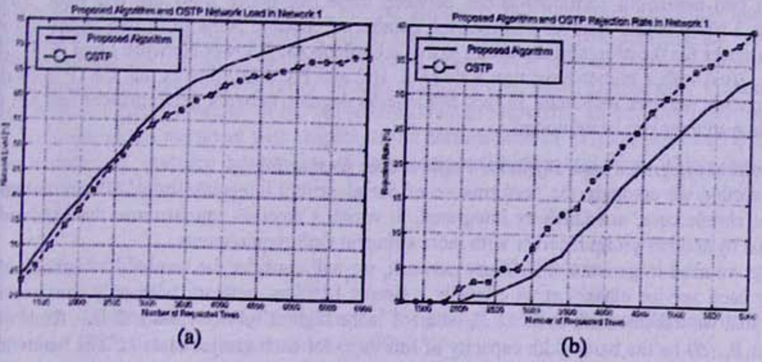

Fig. 1. Proposed Algorithm and OSTP Rejection Rate and Network Load in Network 1

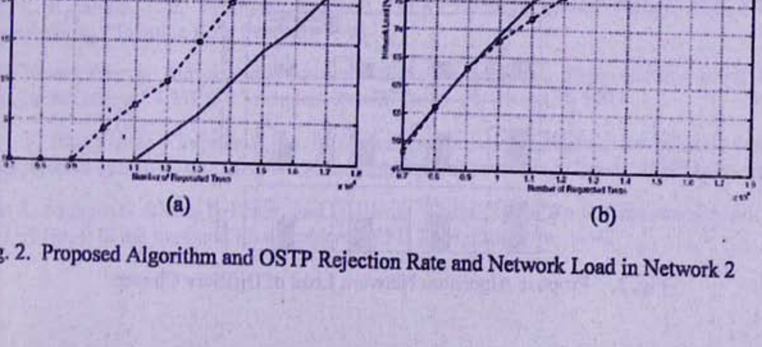

Fig. 2. Proposed Algorithm and OSTP Rejection Rate and Network Load in Network 2

Fig. 1(a) shows that proposed algorithm Rejection Rate in Network 1 is lower than the Rejection Rate of the Optimal solution of the Steiner Tree Problem (OSTP) with $c_{uv}=1$. The Rejection Rate is approximately the same until the Number of Requested Trees is less than 2500. When the Number of Requested Trees overcomes this threshold, the algorithm has a lower Rejection Rate. Both proposed algorithm and OSTP Rejection Rates grow at the growing of the Number of Requested Trees because the same bottlenecks occur. These bottlenecks depend on the network topology and cannot be avoided, but the proposed algorithm Rejection Rate is lower because bottlenecks occur later. Fig. 1(b) shows that our algorithm's Network Load is lightly lower than the OSTP Network Load until the Rejection Rate is the same. Then, since the proposed algorithm rejects fewer trees, its Network Load is higher than the OSTP one.

Network 2 has a higher number of links than Network 1, according to probability function (10) and the used network parameters. In this second type of network, the proposed algorithm Rejection Rate is significantly lower than the OSTP Rejection Rate (Fig. 2(a)). This is because less unavoidable bottlenecks exist. An unavoidable bottleneck that depends on the network topology and not on the algorithm used in the network. For example, if only one path exist between two nodes, all communications between these two nodes must use this path. Since Network 2 has a higher number of links, the number of possible paths from two nodes grows, and it is easier for the proposed algorithm to avoid bottlenecks. A lower Network Load (Fig. 1(b) and Fig. 2(b)) with a higher Rejection Rate (Fig. 1(a) and Fig. 2(a)) proves that OSTP does not use efficiently network resources. In fact, bottlenecks degrade network performances and not all available resources can be exploited.

C. Algorithm integrated with DiffServ: Performance Evaluation

In this section we compare the performance of our algorithm integrated in a DiffServ network with the simple case, not DiffServ integrated, in which a receiver cannot reuse the bandwidth exploited by another group receiver with more stringent QoS requirements.

Since we need to generate a DiffServ network, we will consider the bandwidth capacity of a link for each service class. Let us consider a simple DiffServ network with only four service classes that we indicate with *A, B, C, D*, where *A* is the highest service class and *D* is the lowest one. Let $B_{uv}(cl)$ be the bandwidth capacity of link (u,v) for each service class cl. The bandwidth capacity $B_{uv}(cl)$ for each service class on link (u,v) is randomly generated by using an uniform distribution with mean bandwidth $\overline{B}_{cl}$. For this simulation campaign, a one hundred node network has been randomly generated, with parameters $\alpha=0.2$ and $\beta=0.4$. The link bandwidth capacity for each service class$cl \in \{A, B, C, D\}$ is uniformly distributed with mean bandwidth $\overline{B}_{cl}$ equal to 25Mbps for each service class. We consider the Network Load as defined in (12) of each service class when fifty source-rooted trees, randomly generated, are built by the algorithm. Each receiver asks for a rate uniformly distributed in the range [0,1,2] Mbps, and a service class randomly selected within the set $\{A, B, C, D\}$.

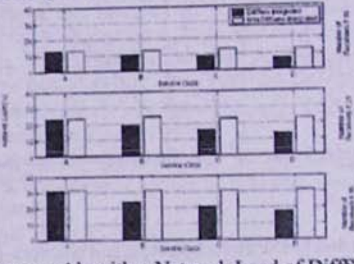

Fig. 3. Propose Algorithm Network Load of DiffServ Classes

For each simulation several runs have been done to ensure a sufficient small trust region and row 95% confidence intervals. Fig. 3 shows that proposed algorithm integrated with DiffServ, performs better than its non integrated version. Furthermore, the Network Load decreases with the service class. In particular, the lowest service class *D* has the lowest Network Load because D class receivers can reuse bandwidth already exploited by all other receivers. Proposed algorithm integrated with DiffServ, performs better than its non QoS-aware version, and this gets evident at the growing of the number of receivers. Fig. 3 shows that for 30 receivers the Network Load for service classes *B*, *C*, and *D* is significantly lower than the Network Load relative to these service classes when this algorithm is not DiffServ integrated (the Network Load is reduced respectively by 23%, 33% and 40%).

4. Conclusion

The presented multicast algorithm aims to effectively avoid bandwidth bottlenecks and to achieve an efficient distribution of link loads in the source-routed tree computation. Seamlessly integrated into the DiffServ Quality of Service approach, this algorithm efficiently manages to address the network partition problem, one of the main reasons for low performance in network, thus increasing the number of multicast communications, as it has been described and tested in an integrated DiffServ aware network through extensive C++ based simulations. The simulation results show a better leverage of the network bandwidth resources, an improved QoS perceived by Multicast group members with respect to the performance of existing multicast algorithms, and a time and resource saving due to low computational complexity, which dramatically characterize the algorithm.

References

[1] L. H. Sahasrabuddhe and B. Mukherjee, "Multicast Routing Algorithms and Protocols: A Tutorial," IEEE Network, January/February 2000.

[2] J. L. Gross and J. Yellen, "Handbook of Graph Theory," CRC Press, Discrete Mathematics and Its Applications, Volume: 25, Series Editor K. H. Rosen, 2003.

[3] R. T. Wong, "A dual ascent approach for Steiner tree problems on a directed graph," Mathematical Programming, 28:271-287, 1984.

[4] H. Takahashi and A. Matsuyama, "An approximate solution for the Steiner problem in graphs," Math, Japonica 6, (1980) 573-577.

[5] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, "Network Flows: Theory, Algorithms, and Applications, "Prentice Hall, February 1993.

[6] Bhattacharyya, Kurose, and Towsley, "Efficient Multicast Flow Control using Multiple Multicast Groups," CMPSCI Technical Report TR 97-15, March 10 1997.

[7] Y. Birk and, D. Crupnicoff, "A Multicast Transmission Schedule for Scalable Multi-Rate Distribution of Bulk Data Using Non-Scalable Erasure-Correcting Codes," IEEE Infocom 2003.

[8] K. Nichols, S. Blake, F. Baker, and D. Black. "Definition of the Differentiated Services Field (DAS Field) in the Ipv4 and Ipv6 Headers," RFC 2474, December 1998.

[9] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," RFC 2475, December 1998.

[10] C. Cheng, R. Riley, S. P. R. Kumar, and J. J. Garcia-Luna-Aceves, "A loop-free Bellman-Ford Routing Protocol without bouncing effect", in ACM Sigcomm '89, pages 224-237, September 1989.

[11] B. M. Waxman, "Routing of multipoint connections," IEEE J. Selected Areas Commun. 6(9) (1998) 1617-1622.

[12] K. Calvert, M. Doar, and E. Zegora, "Modeling Internet Topology," IEEE Communications Magazine, June 1997.

[13] A. Claus and N. Maculan. "Une Nouvelle Formulation du Probleme de Steiner sur un graphe," Technical Report 280. Centre de Recerche sur les Transports, Universite de Mọntreal, 1983.

[14] R. Fourer, D. M. Gay, and B. W. Kernighan. "AMPL: A Modeling Language for Mathematical Programming," Duxbury Press / Brooks / Cole Publishing Company, 2002.

[15] www.cplex.com

Մուլտիմեդիա մուլտիկաստինգներում սպասարկման համար ցանցի հնարավորությունների օպտիմալացում

Վ. Աղազարյան, Հ. Պեդրամ

Ամփոփում

Հոդվածում առաջարկված է լավ կազմակերպված եղանակ աղբյուրի մուլտիկաստ ծառի կառուցման համար: