

A NEW APPROACH FOR BANDWIDTH-DELAY BASED MULTICAST ROUTING ALGORITHMS

MOHAMED Aissa
Shaqra Community Faculty
P.O. BOX 1011 - P.C. – 11961 Shaqra Kingdom
of Saudi Arabia
Email: moha_aissa@yahoo.com

ADEL Ben Mnaouer
L'Ecole Supérieure de Technologie et
d'Informatique
7th November University,
Tunis, Tunisia
Email: abmnaouer@ieee.org

Abstract

New multimedia applications provide guaranteed end-to-end quality of service (QoS) and have stringent constraints on delay, delay-jitter, bandwidth, cost, etc. The main task of QoS routing is to find a route in the network, with sufficient resources to satisfy the constraints. Most multicast routing algorithms give priority to bandwidth minimization during the path computing process. Consequently, paths with end-to-end delays too close to the delay constraint are obtained. During load variation in the network, such paths are prone to delay constraints' violation. In this paper, we first analyze some basic path' establishment methods with bandwidth constraint and then we propose a new approach for the QoS routing problem addressing the tradeoff between bandwidth minimization and the risk level regarding the delay constraint. We propose a fast and simple heuristic algorithm named the Delay-Bandwidth-Inversion Shortest Path (DBSP) algorithm, which attempts to strike a balance between the different QoS parameters. In addition, the proposed algorithm uses information only from neighbouring nodes as it proceeds, which makes it more practical from an implementation point of view. The problem of computing such a multi-constrained multicast tree is NP complete.

Keywords – multicast routing algorithms, end-to-end quality of service, delay-constrained routing, bandwidth reservation.

1. Introduction

Integrated network services are designed to provide quality-of-service (QoS) guarantees to various applications such as audio, video, and data. Many of these applications have multiple QoS requirements in terms of bandwidth, delay, delay-jitter, etc. One of the main problems in QoS-based service routing is how to determine a route satisfying multiple constraints while simultaneously achieving efficient utilization of network resources [1].

Many existing multicast routing algorithms suffer the drawback that the source node must maintain global cost information for the entire network. This may be impractical from an implementation point of view. Therefore, it is important to have the ability to construct near optimal Steiner trees without using global information [2]. Thus, we try to present a heuristic algorithm to construct a multicast tree using local information at each node. That is, the algorithm uses cost information only from neighbouring nodes as it proceeds. Many QoS routing schemes consider only bandwidth and hop count. Hop count is additive, but bandwidth is concave, so these routing problems fall into the category of polynomial time composite problems [3].

Wang and Crowcoft propose several algorithms for QoS routing [4], one of which is a centralized generic source routing algorithm, *the shortest widest-path (SWP)*. In *SWP*, the metrics of interest are bandwidth and delay. It selects the path with largest available bandwidth. If several paths exist with a large bandwidth, the one with the smallest hop count is selected. However, a wider path may achieve higher bandwidth saturation [5].

In [6], Apostopoulos and Williams proposed the *widest-shortest path (WSP)*, which selects the minimum hop count path among those that satisfy the bandwidth requirements. If there are several paths with the same hop count, the widest one (i.e., the one with most available bandwidth) is selected. The problem of WSP is that it selects only one path according to the first metric, no other path can be selected [7].

In [8], Ma and Steenkiste introduced *the shortest-Distance path (SDP)*, also called the *bandwidth-inversion shortest path algorithm (BSP)*. It selects the path with the shortest distance. The distance of a link is defined as the inverse of the available bandwidth on the link, and the distance of a path is the sum of distances over all the links along the path. The distance is defined so that it favors

shortest paths when the load in the network is heavy, and widest paths when the load is medium [9]. It is defined as:

$$\text{dist}(P) = \sum_{e \in P} \frac{1}{B(e)} \quad (1)$$

where $B(e)$ is the bandwidth of the link e and P is the unique path in T (a tree rooted at the source node and spanning all the set of destination nodes) from the source to the end of the link e .

Unfortunately, because of its use of a main cost metric that depends on information about available resources, the SDP requires constantly up-to-date state information. Otherwise considerable performance degradation is observed [10]. To resolve the SDP drawbacks, we propose a simpler algorithm named the DBSP algorithm, which solved the above problem by using *simultaneously* two cost metric functions.

In [12], the *extended-shortest-distance path (ESDP)* was proposed, where the link distance was defined as follows:

$$\text{dist}(P, n) = \sum_{e \in P} \frac{1}{(B(e))^n} \quad (2)$$

where the variable n can be used to tune the algorithm, i.e., by changing n , we can cover the spectrum between shortest ($n=0$) and widest ($n \rightarrow \infty$) path algorithms. The results of experiments in [12] showed that $n=1$ provides the best average throughput for the studied max-min fair share networks.

In [5], Wang and Klara proposed the *enhanced Bandwidth-inversion Shortest Path (EBSP)* algorithm. The *EBSP* is an enhancement proposed to the SDP algorithm, which makes modifications to limit hop count. The authors proposed a *penalty* associated with the hop count of a path that can be used to prevent it from becoming too long. The weight function for a path $P(v_1, v_n) = \langle v_1, v_2, \dots, v_n \rangle$ is defined as follows:

$$\text{dist}(P) = \sum_{j=1}^k \frac{2^{j-1}}{B(e_j)} \quad (3)$$

It was shown in [5], through simulations, that for a simple homogeneous network the *WSP* algorithm is preferred because of its stability. However, for a more complex heterogeneous network, like *DiffServ* networks, the *EBSP* is better since it yields much higher values of the saturate bandwidth.

In [13], the authors proposed the *dynamic-alternative path (DAP)*. It uses the widest-shortest path algorithm but limits the hop count to $n+1$, where n is the minimum hop count in an unpruned network. If no feasible minimal hop path is found, it selects the widest path that is no more than one hop longer. If no such feasible path exists, the flow's request for a path with QoS guarantees is rejected.

It was argued in [8], that the above mentioned algorithms may select a path that is not feasible, either because of stale routing information or because of changes in the network state while the connection is being established. If that happens, the request is rejected. Similarly, it is possible that the algorithms do not find a feasible path, although one exists.

A common weakness of the QoS routing approaches presented above (especially *WSP*, *SWP* and *DAP*) is that they apply in some way or another, a strict priority order when minimizing different QoS parameters. This stringent priority order leads to the construction of unbalanced paths. These paths are optimal in terms of the first considered QoS parameter but inefficient in terms of any subsequently considered QoS parameters. Our contribution extends previous works by providing a new approach for the QoS routing problem. Rather than using a defined priority order between different QoS parameters, we propose a fast and simple heuristic algorithm named the *Delay-Bandwidth-Inversion Shortest Path (DBSP)* algorithm, which attempts to strike a balance between the different QoS parameters by using delay and bandwidth metrics *simultaneously* in the network to create a distance metric function that is derived using both metrics.

In the remainder of this paper, Section 2 will present our network model and the problem definition. Section 3 formally defines the DBSP algorithm along with its analysis. Section 4 includes a comparative study with some well-known algorithms from the literature. In Section 5, we prove the correctness and time complexity analysis of DBSP algorithm. Finally, conclusions are given in Section 6.

2. Our Analytical Model and Problem Definition

2.1. The DBSP Analytical Model

The intuition behind our proposed analytical model is derived from the following observations. *Firstly*, it was shown in [4] that efficient algorithms exist for finding a path subject to the constraints on bandwidth and delay metrics. Therefore, as far as complexity is concerned, bandwidth and delay are natural candidates as routing metrics. *Secondly*, we are keen to propose an algorithm that is using delay and bandwidth metrics *simultaneously* in the network to create a distance metric function that is derived using both metrics. *Thirdly*, we are also motivated by the merits of the *SDP* mentioned in Section 1. Hence, the distance function is based on the delay metric and influenced by the bandwidth metric. Consequently, we define the distance of a link $e = (u, v)$ as follows:

$$\text{dist}(e) = \frac{D(e)}{B(e)} \quad (4)$$

where (as defined before) $B(e)$ - the available bandwidth on the link and $D(e)$ - the delay on the link e . The distance of a path P from the source s to a node v is defined as follows :

$$\text{dist}(P(s, v)) = \sum_{e \in P(s, v)} \frac{D(e)}{B(e)} \quad (5)$$

2.2 Problem Definition

Given a directed asymmetric, acyclic graph $G = (V, E)$ with V as the set of vertices (i.e., the network nodes) and E as the set of edges representing physical or logical connections between nodes. We define the following weight functions on the edges, for any edge $e \in E$:

$D(e): E \rightarrow \mathbb{R}^+$, a positive real delay edge function, and
 $B(e): E \rightarrow \mathbb{R}^+$, a positive real bandwidth edge function.

The delay of a link is the sum of the perceived queuing delay, transmission delay, and propagation delay over that link. The nodes represent routers or switches and edges represent the communication links between them. An edge $e \in E$ from $u \in V$ to $v \in V$ is represented by $e=(u,v)$.

On this graph, we designate a source node $s \in V$ and a set of destination nodes Z , called the multicast group members such that $Z \subseteq V - \{s\}$.

Each link is bi-directional, i.e., the existence of a link $e = (u, v)$ from node u to node v implies the existence of another link $e' = (v, u)$ for any $u, v \in V$. As the network is asymmetric, it is possible that $B(e) \neq B(e')$ and $D(e) \neq D(e')$.

The bandwidth we are interested in here, is the residual bandwidth (some percentage of bandwidth) that is available (reserved) for a new traffic flow (i.e., QoS flows). We define the bandwidth of the unique path $P(s, v)$ in T considered from s to v , as the minimum of the residual bandwidth of all links on the path or the bottleneck bandwidth, and is formulated as follows:

$$BW(e) = \min_{e \in P(s,v)} B(e) \quad (6)$$

Given a delay tolerance (delay bound) Δ , the delay-constrained multicast routing problem can be stated as follows. Find a tree T ($T \subseteq G$) rooted at s and spanning all nodes in Z , such that for each node v in Z , the delay on the path from s to v is bounded above by Δ :

$$Delay[v] = \sum_{e \in P(s,v)} D(e) < \Delta, \forall v \in Z \quad (7)$$

and such that

$$dist(P(e)) = DB(T) = \sum_{e \in T} \frac{D(e)}{B(e)} \text{ is minimized} \quad (8)$$

Delay is additive, but bandwidth is concave, so this routing problem falls into the category of polynomial time composite problems [3]. Therefore, heuristics must be adopted to find sub-optimal approximate solutions.

Next, we give a formal definition of our proposed DBSP algorithm.

3. The proposed Delay-Bandwidth-Inversion Shortest Path Algorithm (DBSP)

Here, we first explain the intuition behind using the Prim-Dijkstra tradeoff algorithm on which our DBSP

algorithm is based and then we clarify the Prim-Dijkstra tradeoff. Subsequently, we explain why in our algorithm, we did not use a tree pruning process. Thereafter, we give the pseudo-code of the DBSP algorithm.

By using longer paths, the SDP algorithm consumes extra resources, and this can block future requests. In [10], it was shown that this is not a crucial issue when the network is light. However, we tried to solve this problem for general cases. Our proposed DBSP algorithm is based on Prim-Dijkstra tradeoff algorithm [13] [14] [15][16][17]. We avoid long paths by giving priority to destination nodes [13]. Our algorithm dynamically adjusts its tree construction policy according to how far a destination node is from the delay bound; the farther the destination node is away from the delay bound, the more priority it has in being selected as parent of the newly added node. Hence, next, we briefly explain the Prim-Dijkstra tradeoff algorithm.

3.1 Brief Overview of Prim-Dijkstra's Tradeoff Algorithm

DBSP is based on Prim-Dijkstra tradeoff algorithm using a greedy strategy based on the shortest-path and minimal spanning trees. It combines the minimum cost and the minimum radius objectives by combining respectively optimal Prim's and Dijkstra's algorithms. It biases routes through destinations [13][14][15][16]. Besides, it uses cost information only from neighboring nodes as it proceeds, which makes it more practical, from an implementation point of view. The min-cost and the min-radius objectives can be respectively addressed by Prim's minimum spanning tree (MST) algorithm [18] and Dijkstra's shortest-path tree (SPT) algorithm [19].

Given a directed asymmetric, acyclic graph $G' = (V, E)$ with V as the set of vertices (i.e., the network nodes) and E as the set of edges representing physical or logical connections between nodes. We define the following weight function on the edges, for any edge $e = (u, v) \in E$:

$C(e) : E \rightarrow \mathbb{R}^+$, a positive real edge cost function

We define the cost on the path from s to v as follows:

$$Cost[v] = \sum_{e \in P(s,v)} C(e), \forall v \in V - T \quad (9)$$

Prim's MST algorithm begins initially with a tree consisting only of a source s . The algorithm iteratively adds an edge $e = (u, v)$ and a node v to T , where u and v are chosen to minimize:

$$C(u, v) \text{ s.t. } u \in T, v \in V - T \quad (10)$$

Dijkstra's SPT algorithm begins with the trivial tree consisting only of the source. The algorithm iteratively adds the edge $e = (u, v)$ and the node v to T , where u and v are chosen to minimize:

$$Cost[u] + C(u, v) \text{ s.t. } u \in T, v \in V - T \quad (11)$$

The algorithm ALG1 (Prim-Dijkstra Tradeoff) proposed in [17] iteratively adds the edge $e = (u, v)$ and the node v to T , where u and v are chosen to minimize:

$$kCost[u] + C(u, v) \quad s.t. \ u \in T, v \in V - T \text{ and } 0 \leq k \leq 1 \quad (12)$$

Here, k is an indicator function. When $k=0$, ALG1 is identical to Prim's algorithm and constructs trees with minimum cost. As k increases, ALG1 constructs a tree with higher cost but lower radius. When $k=1$, ALG1 is identical to Dijkstra's algorithm and constructs trees with minimum radius.

Next we give our modified version of the indicator function (k) as defined in [13].

3.2 Formal Definition of the DBSP

Indicator Function Definition

In our DBSP algorithm, we set the indicator function $k = I_D(u, v)$, which is defined in [13] as follows:

$$I_D(u, v) = \begin{cases} Delay[u]/Delay[u] + Delay[v] & \text{if } u \in Z \\ 1 & \text{otherwise} \end{cases} \quad (13)$$

Where $v \in adj[u]$, i.e., v is a new node not-on-tree and connected (neighbor, adjacent) to the node u , $Delay[u]$ is the sum of the delays of the set of links in T from the source s to the node u , $Delay[v]$ is the total delay of the path in T , emanating from the source to the node v and going through u . Z is the multicast group.

The implication of the above equation is that the farther the destination node's delay is away from the delay bound, the more priority it has in being selected as parent of the newly added node [13].

Now we explain why in our algorithm, we did not use pruning process. Thereafter, we give the pseudo-code of the DBSP algorithm.

One of the drawbacks of the WSP algorithm is that when limiting the hop count, we can only use links belonging to shortest paths. Consequently, we can not fully take advantage of existing longer and lightly loaded routes. For example, this problem was solved in [4] [6], by simply pruning links with insufficient resources first, and then, by calculating the min-hop path over a reduced topology. However as it was observed in [20], when the state information is highly imprecise it is difficult to decide whether or not a link has enough resources to accommodate a new request. The pruning process can remove *infeasible as well as feasible* links, so it degrades the overall performance. Therefore, in this paper, we assume that the pruning process is disabled.

Next we give the pseudo-code of the proposed DBSP algorithm that is depicted in Fig. 1. Lines 1-3 show the initialization and lines 4-14 show the tree construction phase.

3.3 Algorithm Description

The pseudo code of the DBSP algorithm is depicted in Fig. 1. The algorithm repeatedly selects the vertex with

the minimum key value $DB[u] = Delay[u]/BW[u]$. The key represents the least-ratio path estimate from either the source or the destinations found so far. The selected vertex is inserted into S , which is a subset of the vertices whose path from the source has already been determined. We maintain a priority queue Q that initially contains V , and after the extraction operation will contain all the vertices in $V-S$. After each step execution, Q is updated so the nodes are sorted by their values of $DB[u]$. The lowest ratio has the highest priority. We repeatedly select the highest priority vertex from Q . Then, we relax (step 9, 10 and 11) all of its outgoing edges using the incident link ratio, the current ratio estimate of the neighboring node and the indicator function. A pointer array $\pi[v]$ points to the parent node of v in the multicast tree. The path is stored using the variable π . $\pi[v]$ keeps the immediate preceding vertex (called predecessor) of v on the path. Hence, the path can be recovered by tracing the variable π starting from a node, through all intermediate vertices, till reaching the source s .

If all destination nodes are connected, there is no need for looping even the priority queue was not yet empty. This situation is ensured by line 4 (While $Q \neq \phi$ and $Dest \neq \phi$ do).

The condition "*if* $v \notin T$ " in line 9 ensures that cycles in the tree are avoided by preventing consideration of an adjacent node whose route has already been determined.

4. Comparison with Other Algorithms

To compare the algorithms mentioned in Section 1 with DBSP algorithm, we have computed their distance functions on the original graph depicted in Fig. 2. Subsequently, we have applied Dijkstra's shortest path algorithm [19] and then we have pruned all non destination leaf nodes. Examples in Fig. 4 are provided to show the results of the application of these algorithms on the original graph of Fig. 2.

It is worth to mention that during the WSP tree construction process, the second metric was not used. Consequently, the DAP graph is similar to the WSP graph. Based on these graphs, we constructed Table 1, which provides a comparison of our algorithm to some well-known algorithms that are mentioned in Table 1. The average delay in this table is calculated as follows:

$$Average\ Delay = \sum_{u \in Z} Delay[u] / |Z| \quad (16)$$

It can be noted that the *DBSP* tree-construction process performs as good as the best well-known trees indicated in Table 1. We believe that our algorithm stands even better than all the algorithms mentioned in Table 1 as in contrast to the majority of the mentioned algorithms, the second metric was used *during all the tree construction process and simultaneously with the first metric*.

5. Correctness Proof and Time Complexity Analysis of the DBSP algorithm

5.1 Correctness Proof

The correctness of the algorithm DBSP results from the following two theorems:

Theorem 1

The path generated by the DBSP algorithm is loop-free.

Proof

The proof is obvious and results from the fact that both Prim and Dijkstra's algorithms (used in the DBSP algorithm) are loop free. Furthermore, *every node is visited only once* during the construction process. \square

Theorem 2

The algorithm DBSP always constructs a delay-bounded multicast tree if such a tree exists.

Proof

Since DBSP only modifies the cost function used in Dijkstra's and Prim's algorithms, then the correctness in general, results directly from these algorithms. \square

5.2 Time Complexity Analysis

In the following, we analyze the time complexity of our proposed DBSP algorithm.

Theorem 3

The time complexity of DBSP is $O(|E|\log|V|)$.

Proof

Since DBSP only modifies the cost function used in Dijkstra's algorithm and does not affect its asymptotic running time, the time complexity of steps 1-14 of DBSP (Fig. 1), i.e. the tree construction phase is $O(|E|\log|V|)$ assuming that the priority queue Q is implemented as a binary heap (Some improvement can be realized if *Fibonacci* heaps are used for the priority queue [21]). \square

Table 2 shows that our DBSP algorithm has a complexity that is comparable to others well-known algorithms.

6. Conclusion

In this paper, we have first analyzed some basic path establishment methods with bandwidth constraint, then we have introduced a new approach for delay-constrained routing. In contrast to classical approaches, which give absolute priority to delay or bandwidth minimization, the proposed approach tries to strike a balance between minimizing the available bandwidth and meeting the delay constraints of the requested connections. We have presented the Delay-Bandwidth-Inversion Shortest Path (DBSP) algorithm that implements our approach using a simple and efficient distance function. Besides, the proposed algorithm uses delay and bandwidth information

only from neighboring nodes as it proceeds, which makes it more practical from an implementation point of view. It was shown that our proposed DBSP algorithm performs as good as the best well-known algorithms (such as the WSP, SWP, SDP, ESDP, DAP and EBSP). In addition we believe that DBSP is well suited to actual implementation in networks requiring support for multicast communication.

References

- [1] C. Casetti *et al.*, A New Class of QoS strategies Based on Network Graph Reduction, *Proc. Conf. Comp. Commun., (IEEE INFOCOM'02)*, June 2002.
- [2] S. Nelakuditi, Z. Zhang, and R. Tsang, Adaptive Proportional Routing: A Localized QoS Routing Approach, *Proc. Conf. Comp. Commun. (IEEE INFOCOM'00)*, Mar. 2000.
- [3] Ilmari Juva, Analysis of quality of service routing approaches and algorithms, *Master's thesis, Helsinki University of technology*, Marsh 2003.
- [4] Z. Wang and J. Crowcroft, Quality-of-Service routing for supporting multimedia applications, *IEEE JSAC*, 14(7):1288-1234, September 1996.
- [5] J. Wang and Klara Nahrstedt, Hop-by-hop routing algorithms for premium-class traffic in DiffServ Networks, in *Proceedings of IEEE Infocom2002*, June 2002.
- [6] G. Apostopoulos, D. Williams, and others, QoS routing mechanisms and OSPF extensions, *RFC2676*, 1999.
- [7] H. Jeon, S. kim, et all, Performance analysis of QoS Routing with network dependant delay cost, *Proceedings of the 15 International Conference on Information Networking (ICON'01)*, IEEE 2001.
- [8] Q. Ma and P. Steenkiste, On path selection for traffic with bandwidth Guarantees, in *Proceedings of IEEE International Conference on Network Protocols*, pp. 191-202, October 1997.
- [9] X. Xiao and M. Ni, Internet QoS, A Big Picture, *IEEE Network*, vol. 13, No 2, pp. 8-18, March 1999.
- [10] Karol Kowalik and Martin Collier, Should QoS routing algorithms prefer shortest paths?, *IEEE 2003 International Conference on Communications*, vol. 1, Anchorage, Alaska, May 2003, pp.213-217.
- [11] Q. Ma and P. Steenkiste, Routing high-bandwidth traffic in max-min fair share networks, *ACM SIGCOMM'96 Stanford*, pp. 206-217, August 1996.
- [12] Q. Ma and P. Steenkiste, Quality-of-service routing for traffic with performance guarantees, in *IFIP fifth International Workshop on Quality-of-service*, pp. 115-126, May 1997.
- [13] Mohamed Aissa and Adel Ben Mnaouer, A New Delay-constrained Algorithm for Multicast

- Routing Tree Construction, *Intl. Jour. of Comm. Systems, IJCS*, pp.985-1000, Sept. 2004.
- [14] A. Shaikh and K. Shin, Destination-Driven routing for low-cost Multicast, *IEEE J.Select. Areas Commun.*, 15:373-381, April 1997.
- [15] A. Shaikh, S. Lu, and K. Shin, Localized multicast routing, in *Proc. Of IEEE GLOBECOM*, (Singapore), pp. 1352-1356, November 1995.
- [16] L. Guo and I. Matta, QDMR, An efficient QoS Dependent Multicast Routing Algorithm, *J. of Communications and Networks-Special Issue on QoS in IP Networks*, 2(2), June 2000.
- [17] C. J. Alpert, T. C. Hu, J.H Huang, A direct combination of the Prim and Dijkstra constructions for improved performance-driven global routing, *Proc. IEEE Intl. Symp. On Circuit and Systems*, Chicago, May 1993, pp. 1869-1872.
- [18] B. C. Prim., Shortest connecting Networks and Some Generalizations, *Bell System Tech. Journal*, 36 (1959), pp1389-1401.
- [19] E. W. Dijkstra, A Note on Two Problems in Connection With Graphs, *Numerische Mathematik*, 1 (1959), pp. 269-271.
- [20] A. Shaikh, J. Rexford and K. G. Shin, Evaluating the impact of stale link state on quality-of-service routing, *IEEE/ACM Transactions on Networking (TON)*, vol. 9, no.2, pp. 162-176, 2001.
- [21] S. Saunders, T. Takaoka, Improved shortest path algorithms for nearly acyclic graphs, *Theoretical Computer Science*, v. 293(3): pp535-556, February 2003.

INPUT : $G=(V,E)$: a directed, asymmetric and acyclic graph (the network topology), s = source node, Z : Multicast group ($Z \subset V$), Δ : Delay bound, $D(\cdot)$ = Link delay function, $B(\cdot)$ = Link bandwidth function, S : A subset of the vertices whose path from s has been computed (determined).

OUTPUT: A delay bounded Steiner tree T spanning $Z \cup \{s\}$ and rooted at s . /* $\pi[u]$ is a pointer to u 's parent in T - Q is a priority queue */

DBSP ($G(V,E)$, s , Z , Δ , D , B)

```

1   $DB[s]=0, Delay[s]=0, T = \phi; S = \phi; Q \leftarrow V; Dest \leftarrow Z;$ 
2  For each vertex  $u \in V - \{s\}$  do
3
4  While  $Q \neq \phi$  do and  $Dest \neq \phi$  /* the loop is executed as long as Q is not empty and all destination nodes are not yet connected */
5       $u \leftarrow Q$  /* pick from Q the next not-on-tree node with minimum cost */
6       $T = T \cup \{u\}$ 
7      For each vertex  $v \in Adj[u]$  do /* for each not-on-tree node v adjacent (neighbour) to u */
8          if  $Delay[u] + D(u, v) < \Delta$  AND
9              if  $DB[v] > I_D(u, v)DB[u] + D(u, v) / B(u, v)$  AND  $v \notin T$ 
10                  $DB[v] \leftarrow I_D(u, v)DB[u] + D(u, v) / B(u, v)$ 
11                  $\pi[v] \leftarrow u;$ 
12                  $Q \leftarrow Q - \{u\}; S \leftarrow S + \{v\};$ 
13                 if  $v \in Z$ 
14                      $Dest \leftarrow Dest - \{v\};$ 

```

Fig .1 :The Proposed DBSP Algorithm

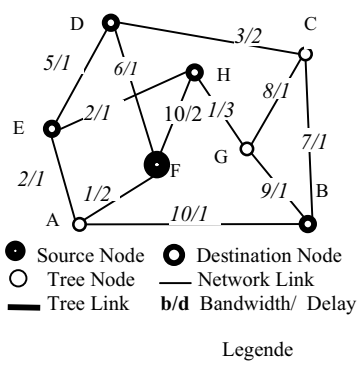


Fig. 2 Original Graph

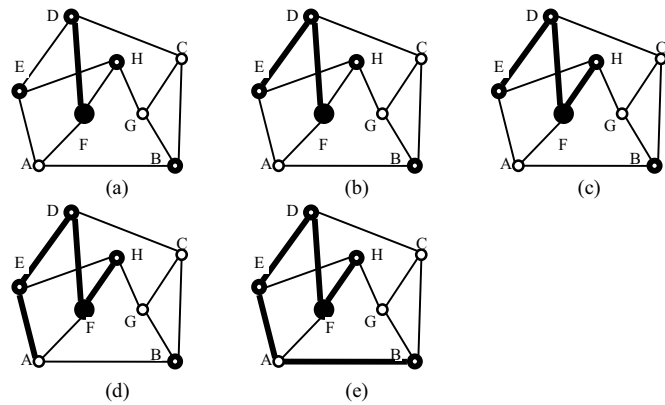


Fig. 3 DBSP Method Execution (Our Method)

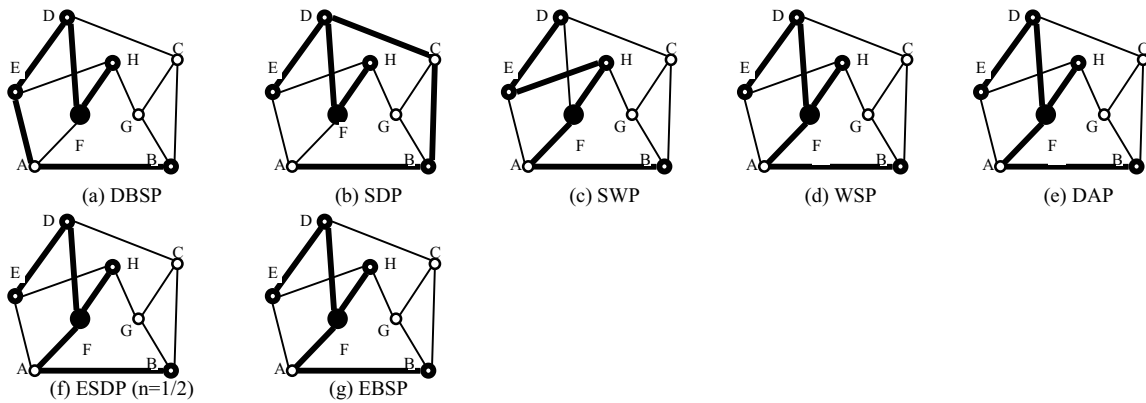


Fig.4 A Summary of all resulting multicast trees

Table1: Comparison between DBSP and Other Algorithms by Delays

Path	1 st / 2 nd Metric	Use of the 2 nd Metric	Number of times the 2 nd Metric is used	Total Delay	Average Delay	Delay Tolerance coefficient	Figure 9
WSP	Delay / Bandwidth	Not used at all	0	8	2	2	(d)
DAP	Delay / Bandwidth	Not used at all	0	8	2	2	(e)
ESDP	Bandwidth	Not used at all	0	8	2	2	(f)
EBSP	Bandwidth	Not used at all	0	8	2	2	(g)
DBSP	<u>Delay & Bandwidth simultaneously</u>	<u>Simultaneously with the 1st metric</u>	5 (*)	9	2.25	1	(a)
SDP	Bandwidth	Not used at all	0	10	2.50	0	(b)
SWP	Bandwidth / Delay	During selection between 2	2	12	3	1	(c)

* -Which is equal to the phase number of the DBSP tree construction process.

Table 2: Algorithm Complexities

Algorithms	Time Complexities
DAP, ESDP, EBSP, SWP, SDP, WSP	$O(E \log V)$
DBSP	$O(E \log V)$