The Bounded Loss Rate (BLR) Problems in Multicast Networks

Part I : Problem Formulation

Deron Liang and Da-Wei Wang

Institute of Information Science
Academia Sinica
Taipei, Taiwan 11529
Republic of China

1. *Introduction*

Multicasting is the simultaneous transmission of data to multiple destinations. Problems of the *multicast routing* in high speed networks, such as ATM networks, have been recognized as important problems because of its application in video conferencing [1] and HDTV broadcasting [2]. The multicast routing problem has usually been formalized as an off-line allocation problem , where the network is normally modeled by the graph representation [4] and the set of multicast streams to be considered is static and given [5][6]. When a multicast stream arrives at the network, the multicast routing algorithm is responsible for finding routes from the source to each of the destinations; each route should have bandwidth available to support the stream. If multiple routes exist for a given multicast stream, the routing algorithm will choose one so as to optimize a certain objective function.

The existing algorithms can be classified into two categories in terms of objective functions: the *shortest-path algorithms* [3]and the *minimum cost algorithms* [4][9]. In the former category, the routes are computed independently form the source to each destination using Dijkstra's shortest path algorithms [3], then the multicast route is merged to form a multicast tree. For the algorithms in the second category, the multicast routes are constructed in such a way that the sum of the costs associated with the used links is minimized. This problem is usually cited as the *Steiner tree* problem, which is known to be NP-complete [4][5]. Numerous heuristic algorithms have been proposed to study this problem [6][7]. Kompella proposes an extension to the Steiner tree problem where the minimum cost is searched with delay constraints [10]. Previous algorithms can handle a single multicast stream at a time, Noronha and Tobagi study a routing problem with batch arrivals and propose an optimal algorithm using integer programming approach [8].

Previous works have focused on the issue of efficient bandwidth allocation. As the new applications emerge, in particular, the multimedia applications, the guarantee of quality service (or QoS) becomes a major concern in designing new generation of routing algorithms. In this paper, we propose a new problem, called *Bounded Loss Rate* problem (BLR), where end-to-end total cell loss rate of each point-to-point connection is taken into consideration when routing a multicast stream in the network. The cell loss property of each communication channel in the network is characterized by a loss rate function that is typically a function of the traffic loading over that link. (See Figure 1 for example.) A new stream usually specifies not only the

minimum bandwidth requirement but also the end-to-end loss rate bound associated with each destination. The loss rate bound may vary from destination to destination. The goal of this research is to develop efficient algorithms to route these multicast streams over the network in such a way that the loss rate bounds are all satisfied for the new stream while the none of the loss rate bounds of the existing streams is violated, provided such a routing exists.

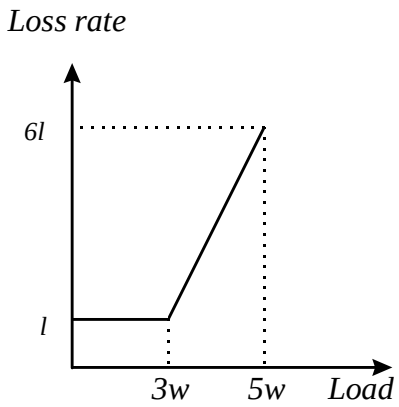

Figure 1. An typical loss rate function of a link.

The organization of this report is as follows. In Section 2, a few of the related works are reviewed. In Section 3, we formally define the BLR problems. In Section 4, the complexity of the BLR problems are analyzed. Finally, the research contribution is concluded in Section 5. The proofs of the NP-completeness of the BLR problems are presented in the appendices.

2. *Related Works*

2.1 The Steiner tree problem

2.1.1 Problem definition:

Input:

1. A simple undirected graph $G = (V, E)$, where V is the set of vertices in G, and E is the set of edges in G.
2. A link-cost function associated with each edge $C(e) : E \rightarrow R^+$.
3. A set of destinations D and $D \subseteq V$.

Goals:

Find a ***Steiner tree*** of G that spans D with minimal total cost on its edges.

2.1.2 KMB heuristic algorithm:

It has been shown that the *minimum cost Steiner tree* (MST) problem is a *NP-complete* problem [4]. The *KMB* heuristic algorithm solves the MST problem with near optimal performance [9].
Here are the 5 steps in the KMB algorithm:

Begin

Step 1. Find the shortest path for each pair of destinations to form a *cost graph* G_1.

Step 2. Find the minimum spanning tree T_1 of G_1.

Step 3. Replace each edge in T_1 by its corresponding shortest path in *G* to form G_S.

Step 4. Find the *minimum spanning tree* of G_S to form T_S.

Step 5. Delete edges in T_S, if necessary, so that all the leaves in T_H are destinations.

End.

The inputs and outputs in and from each step are illustrated in Table 1.

	Input	Output
1	*G, D*	G_1 (cost graph of *G*)
2	G_1	T_1 (MST of G_1)
3	*G*, T_1	G_S
4	G_S	T_S (MST of G_S)
5	T_S	T_H

Table 1. The inputs and outputs in/from each step in the KMB algorithm.

2.2 Multicast Routing for Multimedia Communication [10]

2.2.1 The Constrained Steiner Tree Problem definition:

Input:

1. A simple undirected graph *G* = (*V*, *E*), where *V* is the set of vertices in *G*, *E* is the set of edges in *G* .
2. A source node *s*.
3. A link-cost function associated with each edge $C(e) : E \rightarrow R^+$.
4. A link-delay function associated with each edge $L(e) : E \rightarrow R^+$.
5. A set of destinations *D* and $D \subseteq V$.
6. A bounded delay tolerance Δ.

Goals:

Find a *Constrained Steiner Tree* (CST) *T* such that the total cost $\sum_{e \in T} C(e)$ is minimized; i.e., *T* is the *Minimal Cost Constrained Steiner Tree* (MCST).

Note: a ***Constrained Steiner Tree (CST)*** *T* is a tree in *G*, rooted at *s*, that spans the nodes in *D* such that for each node *d* in *D*, if *P*(*s*, *d*) is the path in *T* from *s* to *d*, $\sum_{e \in P(s,d)} L(e) < \Delta$.

2.2.2 The source-based routing algorithms for the Constrained Steiner Tree(CST):

1. ***Definition:***

- A *constrained cheapest path* between *v* and *w* is the least cost path from *v* to *w* that has delay less than Δ. We denote the cost on such a path by $P_C(v,w)$ and the delay on it by $P_L(v,w)$.
- The *cost of the cheapest path* from *v* to *w* with delay exactly *d*, $C_d(v,w)$, if there are multiple cheapest constrained paths with the same cost, then the one with the least delay is chosen. We can formulate $C_d(v,w)$ and $P_C(v,w)$ as follows:

$$C_d(v,w) = \min_{u \in V} \left\{ C_{d-L(v,u)}(v,u) + C(u,w) \right\}$$

$$P_C(v,w) = \min_{d < \Delta} C_d(v,w)$$

- A *closure graph* G' on a set of nodes *N* is a complete graph on the nodes in *N* with edge cost between nodes *v*, *w*∈*N* equal to $P_C(v,w)$ and edge delay $P_L(v,w)$.

2. ***There are 3 steps in this algorithm:***

Begin

Step 1. In order to compute the closure graph G', we determine the constrained cheapest paths between all pairs of nodes in the set $D \cup \{s\}$. Since $P_L(v,w)$ is determined by the constrained cheapest path that corresponds to $P_C(v,w)$, thus we can construct the closure graph G' on the nodes in the set $D \cup \{s\}$

Step 2. Construct a constrained spanning tree of G'. We use a greedy approach to add edges to a sub-tree of the constrained spanning tree until all the destination nodes are covered. Assume *v* is in the tree constructed thus far, and that we are considering whether to include some edge adjacent to *v*. We have considered the following two selection functions:

$$f_{CD}(v,w) = \begin{cases} \dfrac{C(v,w)}{\Delta - (P(v) + L(v,w))}, & if \ P(v) + L(v,w) < \Delta \\ \infty, & otherwise \end{cases}$$

and

$$f_C = \begin{cases} C(v,w), \text{ if } P(v) + L(v,w) < \Delta \\ \infty \quad , \text{ otherwise} \end{cases}$$

The delay from source to *v* is within the delay bound, where *P*(*v*) is the delay on the path from *s* to *v* in the spanning tree constructed thus far.

Step 3. Expand the edges the constrained spanning tree into the constrained cheapest paths they represent, and remove any loops that may be caused this expansion. The edge selection functions, f_{CD} and f_C, give rise to two *source-based heuristics*: CST_{CD} and CST_C, respectively.

- ***Heuristic*** CST_{CD} uses the selection function f_{CD}, which explicitly uses both cost and delay in its functional form. It tries to choose low-cost edges, but modulates the choice by trying to pick edges that maximize the residual delay. CST_{CD} increases the chances of extending the path through this edge, and beyond to another destination. It reduces the cost of the tree through path sharing, and it also has a tendency to optimize on delay. So it may find paths with delays far lower than Δ, at the expense of added cost to the tree.
- ***Heuristic*** CST_C minimizes f_C, thereby trying to construct the cheapest tree possible while ensuring that the delay bound is met. This tends to minimize the cost of the tree without unduly minimizing the delay.

End.

2.2.3 Conclusion:

A. CST_{CD} and CST_C always produce a constrained spanning tree, if one exists. Moreover, CST_C has better average performance than CST_{CD}. But for distributed algorithms, we find that CST_{CD} works better than CST_C.

B. CST_{CD} performs marginally worse than CST_C. As the multicast group increases in size, the algorithms CST_{CD} and CST_C converge to the minimum spanning tree. It means that the heuristics converge to the optimal solution for large group sizes.

C. If adequate global information is available to the source, then the source-based heuristics produce much better results than shortest delay routing.

2.3 Delay-Bounded Minimum Steiner Tree (DMST) problem [11]

2.3.1 Problem definition:

The problem of ***DMST*** is specified as follows:

Input*:*

1. A simple undirected graph $G = (V, E)$, where V is the set of vertices in G, E is the set of edges in G.
2. A source s .
3. A link-cost function associated with each edge $C(e) : E \rightarrow R^+$.
4. A link-delay function associated with each edge $L(e) : E \rightarrow R^+$.
5. A set of destinations D, $D \subseteq V$.
6. A *DDF* (Destination Delay-bound Function) associated with each destination in D, $\delta(d)$: $D \rightarrow R^+$.

Goals*:*

Find a *Steiner tree* of G that spans $D \cup \{s\}$ so that the cost function of the tree is minimized while *DDF* is satisfied.

2.3.2 The Bounded Shortest Multicast Algorithm (BSMA):

1. Definition:

- A tree obtained during each refinement is called a *tree configuration*, and the *j-th tree configuration* is denoted by T_j .
- *Path switching* means that a path in T_j is replaced by a new path that is not in T_j, resulting in a new tree configuration T_{j+1} .
- T_j' is the *collapsed tree* of T_j, for representing the candidate paths chosen in the path switching. It consists of nodes and *super-edges*, where the set of nodes of T_j' contains source node , destination nodes of T_j and those nodes of T_j that are connected by more than two tree edges in T_j .
- A *superedge* in T_j' is the longest simple path in T_j in which all internal nodes (i.e. excluding the end nodes of this path) are relay nodes and each relay node connects exactly two tree edges.

2. The ***BSMA*** is summarized as follows:

Begin

Step 1. Construct the initial tree T_0 which is a *minimum-delay Steiner tree*, with respect to the multicast source, using *Dijkstra shortest path* algorithm.

Step 2. Then iteratively refines T_0 for low cost. The refinement from T_j to T_{j+1} (initially, j=0) is accomplished by *path switching*. An effective ***delay-bounded path switching*** during the *j-th* refinement involves:

- Choosing the path to be taken out of T_j:

  Deletes a superego from T_j', resulting in two sub-trees T_j^1 and T_j^2, where $T_j = T_j^1 \cup T_j^2 \cup P$ (*P* is the superego that be deleted).
- Selecting the new path in *G* not in T_j that replaces the path to be deleted from T_j:

  Find a *delay-bounded shortest path Ps* to reconnect T_j^1 and T_j^2 .A *delay-bounded shortest path Ps* between T_j^1 and T_j^2 is defined as the path with the smallest cost, subject to the constraint that the new tree $T_{j+1} = T_j^1 \cup T_j^2 \cup Ps$ is a *delay-bounded tree*.

There are two heuristics: ***Path-Switching Heuristic*** and ***Greedy Path-Switching heuristic***.

1. Path-switching heuristic:

A. Initially, all super-edges are *unmarked*.

B. Among all unmarked super-edges, ***BSMA*** selects the superego *Ph* with the *highest path cost*, and exchanges it with another superego such that the resulting paths to destinations are *delay-bounded*.

Note that the delay-bounded shortest path algorithm always terminates, because at least the deleted path is found again.

2. Greedy Path-Switching heuristic:

Definition : Gain is the cost reduction after a round of path-switching. Let *p* be a path in tree T_j with cost *c*, and *P** the corresponding delay bounded shortest path to be added into tree T_{k+1} with cost *c**.

The gain *g* of this path switching is defined as $g = c - c^*$.

A. BSMA computes gains of all pairs of possible path switching in T_j, and then selects the one with the maximum gain.

B. BSMA continues the greedy path switching, and terminates when the maximum gain is zero.

End.

2.3.3 Conclusion:

A. The *Greedy Path-Heuristic* gets the costs of trees within 2% difference but much longer running time than *Path-Switching Heuristic*. So this paper

results are obtained based on *BSMA's Path-Switching Heuristic*.

B. The algorithm can handle two variants of the cost function: For *utilization-driven* multi-casting, the path cost is the *sum* of link costs along his path. For *congestion-driven* multicasting, the path cost is the *maximal* link cost along the path.

C. Instead of using the one-pass growing of the multicast tree, propose an iterative optimization process to further minimize the tree cost.

D. The simulation results show that *BSMA* produces delay-bounded multicast trees that, depending on the delay bounds imposed, can be very close to having minimum cost.

E. The performance comparisons between the *Greedy Path Switching* and *Path-Switching Heuristic* algorithm.

2.4 Degree-Constrained Multi-casting Routing in Point-to-Point Networks [14]

2.4.1 The Degree-Constrained Steiner Problem (DCSP):

DCSP is defined as follows:

Input:

1. A simple, undirected, connected graph $G = (V, E)$, where V is the set of vertices in G, E is the set of edges in G.
2. A source node s.
3. A link-cost function associated with each edge $C(e) : E \rightarrow R^+$.
4. A set of destinations D and $D \subseteq V$.
5. The node degree constraints $k_v \geq 2, \ \forall v \in V$.

Goals:

Find a MST, T in G, such that $d_v \leq k_v \quad \forall v$ in T and $\sum_{e \in T} C(e)$ is minimized.

2.4.2 Algorithms for the MST problem:

A. Unconstrained Steiner Tree Heuristics:

A.1 Heuristic Naive: [13]

It starts with an arbitrary multicast member as the multicast tree. It then repeatedly connects another random multicast member to the multicast tree by the shortest path between the new member and the multicast tree until all the members are in the multicast tree.

A.2 Shortest Path Heuristic (SPH): [16]

It initializes the multicast tree to an arbitrary multicast member. It then joins the next closest multicast member to the multicast tree by the shortest

path between the multicast member and the tree. It terminates when all members have joined the tree.

Note:

SPH is differs from Heuristic Naive because multicast members join in the order determined by their distance to the multicast tree, rather than in random order.

A.3 Heuristic SPH-Z: [16]

This variant of SPH applies the basic SPH algorithm described in above once for each possible choice of the starting Z-node, returning the best solution found.

A.4 Heuristic K-SPH: [17]

The Kruskal-based shortest-path heuristic, starts with the forest of multicast member nodes. It repeatedly joins the two closest multicast member subtrees until a single tree spanning all multicast members remains.

A.5 Heuristic ADH: [17]

Like K-SPH, it starts with the forest of multicast member nodes. It repeatedly connects the three closest multicast member components through the most central node. It terminates when a single tree remains, spanning all multicast members.

A.6 Heuristic Dual Ascent: [18]

The Dual Ascent heuristic finds a solution to the SPN in the following five steps:

I. Convert the undirected graph *G* into a directed graph *G'* by substituting every undirected edge by two directed edges of equal weight in opposite directions.

II. Build a directed subgraph *A* from *G* containing a solution. This step starts with subgraph *A* containing all node nodes of *G*, but none of its edges. Edges are added to subgraph *A* one at a time until at least one multicast node is connected to every other multicast member.

Note:

In directed graphs, node i is connected to node j if a directed path exists from i to j.

III. Convert *A* to an undirected graph *U*.

IV. Find *U*'s minimum spanning tree *T*.

V. Prune all non-multicast member leaves from *T*. The result is the multicast tree.

B. Degree-Constrained Steiner Tree Heuristics: (The detail of the algorithm is not fully explained in the text; in particularly, the handling of degree constraint violation.)

B.1 Modified Steiner Tree Heuristics:

B.1.1 ADH: ADH connects only the closest subtree to the most central node. This is because of the difficulty of connecting greater than two components as explained in above.

B.1.2 Dual Ascent: Modified Dual Ascent differs from it unconstrained equivalent because it does not find the degree-constrained minimum spanning tree to its subgraph. Instead, it uses ***SPH*** to generate a Steiner tree from the subgraph.

B.2 Constrained Heuristics: [19]

B.2.1 A29: It is also a variant of ***SPH***. It first adds enough edges of infinite weight to make *G* a complete graph and then applies heuristic ***SPH***. Since *G* is complete, ***A29*** will always find a degree-constrained Steiner tree, though the solution may contain infeasible edges.

B.2.2 SPH-R: It is our own variation of ***SPH***. ***SPH-R*** like ***SPH-Z*** repeatedly applies SPH to the graph *G* for different starting points. However, ***SPH-R*** terminates the first time it generates a solution.

2.4.3 Conclusion:

A. Many of the Steiner heuristics tested yielded degree-constrained multicast trees within 5% of the best heuristic solution found in almost all the networks tested. And few of our networks were unsolvable, in those cases, backtracking solved many of the remaining cases.

B. Simple Steiner heuristics such as ***SPH*** and ***SPH-R*** emerged as the clear winners with an attractive balance between the conflicting objectives of solution quality and algorithm complexity.

C. The next least expensive heuristics ***K-SPH*** and ***ADH*** often gave better solutions at moderate, extra expense. ***Heuristic Naive***, our expected worst Steiner heuristic, often did produce the worst solution; however, it also produced many solutions of surprisingly high quality.

D. Degree-constrained heuristics easily solved all the dense networks we tested without backtracking.

2.5 How Bad is the Naive Multicast Routing? [13]

2.5.1 Problem definition:

There are two types of multicast groups:

- **Static:** once set-up, remain unmodified until they are discarded or torn down.

- **Dynamic:** destination nodes which are already part of the group may wish to leave or destination nodes which are not part of the group may want to join.

Now we discuss the dynamic case.

Input:

1. A simple undirected graph $G = (V, E)$, where V is set of vertices in G and E is the set of edges in G.
2. A source s.
3. A link-cost function $C(e) : E \rightarrow R^+$.
4. A set of destinations D and $D \subseteq V$.
5. A MST T_t for the specific time t.
6. A set of destinations $D_{t+1} = (D_t - D_{t+1}^-) \cup D_{t+1}^+$ and $D_i \subseteq V \; \forall \; i$, where D_i means the set of destinations at the specific time i , D_i^+ means the set of vertices want to join the destination set at the specific time i and D_i^- means the set of vertices want to leave the destination set at the specific time i .

Goals:

Find *MST* T_{t+1} in G which spans D_{t+1} so that the total cost of T_{t+1} is minimized.

Note: *Nodes in the sub-tree* T_{t+1}*, but not in* D_{t+1} *are called Steiner Nodes.*

2.5.2 The algorithm

- *Node Addition*

  The particular algorithm chosen was to find ***the shortest path*** from the source to the node which had been chosen to be added and to add all the nodes along that path into the multicast connection as Steiner nodes.

- *Node Removal*

  Not discussed in the paper.

2.5.3 Conclusion:

A. Increasing the graph size results in an increase in the inefficiency, but the changing proportion of destinations makes only a minor difference to the inefficiency.

B. Increasing the graph size results in little change in the inefficiency if the degree of nodes $\forall v \in V$ is constant, and the inefficiency is relatively independent of the number of nodes in the network for a given proportion of destinations involved in a multicast connection.

C. The values of the inefficiency obtained with hierarchical model are less than those obtained with the constant degree model, probably due to a more limited set of inter-cluster paths from node to node, so the naive algorithm is

more likely to choose the same ones as an optimal MST algorithm.

D. There is a approximately linear relationship between the degree and the inefficiency. As the degree increases, the number of available paths between any two nodes increases and hence the probability of the shortest path tree coming close to the heuristic KMB tree decreases and the inefficiency increases.

E. The paths chosen by the naive algorithm are indeed shorter than those chosen by KMB and the length of the paths increases as the degree of connectivity of the graph increases. And the maximum path length tends to increase in KMB trees as the proportion of destinations increases.

2.6 Dynamic Multicast Routing Algorithms [12][15]

2.6.1 Problem definition:

Input:

1. An undirected graph (network) $G = (V, E)$, where V = set of vertices in G and E = set of edges in G.
2. A link-cost function $C(e) : E \rightarrow R^+$.
3. A source s.
4. A MST T_t for the specific time t.
5. A set of destinations $D_{t+1} = (D_t - D_{t+1}^-) \cup D_{t+1}^+$ and $D_i \subseteq V \; \forall \; i$, where D_i means the set of destinations at the specific time i , D_i^+ means the set of vertices want to join the destination set at the specific time i and D_i^- means the set of vertices want to leave the destination set at the specific time i .

Output:

Find MST T_{t+1} in G which spans D_{t+1} so that the total cost of T_{t+1} is minimal.

2.6.2 Dynamic multipoint algorithms

1. The greedy algorithm: [20]

- This finds the nearest node already in the multicast tree to the node to be added and connects the two via the shortest path between the two nodes.
- To delete a node from the multicast group, it is first marked as 'deleted' and if that node is not an internal node in the connection, the branch of which it is a part is pruned.

2. The source rooted Shortest Path (SP) algorithm:

I. Choose a node at random and call that the source node of the multicast group.

II. Another different node was also chosen and the shortest path was found between them, forming the initial multicast connection.

III. To add a node to the group, the shortest path from the source node to the node chosen to be added was found and all the nodes along that path were added into the multicast connection as Steiner nodes.

IV. Then took the union of these paths to produce a multicast tree that was resilient to change and had a mean inefficiency within 1.5 times that of the KMB .

3. Geographic Spread Dynamic Multicast (GSDM) Routing Algorithm: [15]

Part 1. Definition

Geographic spread (GS) is defined as follows:

Given a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, and a subset $U \subseteq V$, the ***geographic spread (GS)*** of the set U, in the static case when tree T spans U, is defined as the inverse sum of the minimum distance from a vertex v to a vertex in T, over all vertices $v \in V$.

i.e $GS(U, V, E) \equiv \left[\sum_{v \in V,\, u \in T} \min\{dist(v,u)\} \right]^{-1}$

(*Note*: *If there is more than one path with the least cost, choose the path which maximizes the GS from the set of minimum cost paths and append it to the tree T*).

Part 2. Algorithms

A. Node Addition

Step 1. Select the node to be added that is not already in the multicast group, the set *D*, and call this node *A*.

Step 2. Find the nearest node in the multicast tree *Ts*, to node *A*, and call this node *B*. Node *B* does not necessarily have to be in the multicast group, the set *D*.

Step 3. Find the nearest two nodes to node *A*, which are neighbors on either side of node *B* in the multicast connection or tree *Ts* and call these nodes *C* and *D* respectively. By neighbors, one means that there exists a path from the node in the multicast group, the set *D*, in the multicast tree *Ts*, to node *B* that does not pass through any other node that is in the multicast group, the set *D*. If however, node *B* is a leaf of the multicast tree *Ts*, (i.e. its degree=1), then find node *B* neighbor which lies in the multicast tree, *Ts* and call this node *C*. If there is just one member in the multicast group, the set *D*, (i.e. the source node), just join node *A* to node *B* by the shortest path as node

B does not have any neighbors and it will not be necessary to do steps 4 and 5.

Step 4. If node *B* is in the multicast group, the set *D*, one considers which of three paths is the cheapest from the set of paths:
C-B-D & *B-A* (greedy algorithm)
C-B-A-D
C-A-B-D
If node *B* is not in the multicast group, the set *D*, one then also considers the path *C-A-D*. If node *B* from *Step 3*. is a leaf node, one only has two paths to consider, i.e., path *C-B-A* and path *C-A-B*.

Step 5. If there is more than one path with the least cost, choose the path which maximizes the geographic spread from the set of minimum cost paths and append it to the tree *Ts*.

Step 6. Add node *A* to the multicast group, the set *D*. When all the modifications are done, one's finished.

B. Node Removal

Step 1. Select a node to be removed from the multicast group, the set *D*. The selected node should not be the source node.

Step 2. If the selected node is a leaf node, then remove the node from the multicast group, the set *D*, and prune the branch of which it is a part from the multicast tree or connection. If the selected node is not a leaf node, then mark it as eleted from the multicast group, but do not remove it from the multicast connection, until it is a leaf node in the multicast connection.

Step 3. Repeat step 2 and 3 until all modifications are done.

2.6.3 Conclusion:

A. The mean inefficiency of the ***GSDM*** routing algorithm rises slightly with increasing number of nodes in the graph.

B. The *Greedy's* mean inefficiency does not seem to follow a definite pattern as the number of nodes in the graph is increased from 20 to 100 nodes.

C. The *SP* algorithm's mean inefficiency rises sharply with increasing number of nodes in the graph.

D. For a 50 node random graph, the ***GSDM*** algorithm is not as efficient when nodes are in the multicast group than when there are more nodes in the multicast group. However, as the number of nodes in the multicast group is increased from 10% to 100% of total nodes in the graph, the algorithm

efficiency gets better.

E. ***GSDM*** routing algorithm performs very well in comparison to the *KMB* algorithm with an inefficiency just slightly worse than the near optimal heuristic's when tested on random graphs.

3. *The Problem Formulation*

The following are the terminology used in the definition of the BLR problem:

1. *Network Topology*: is represented by a graph *G* = (*V, E*), where *V* is the set of vertices and *E* is the set of edges.
2. *Loss rate function* $L_e(\varpi)$: associated with an edge *e*, $\forall e \in E$, where ϖ = *load*.
3. *Stream* S is specified by $\{s, \varpi, (d_1, b_1), ..., (d_k, b_k)\}$, where s is the source, ϖ is the expected loading of the stream, and k pairs of (d_i, b_i) where d_i : *i*-th destination and b_i : loss bound associated with d_i.
4. *Multicast tree* $T(S)$: is a Steiner tree in *G* with s as the source, and $d_1, ..., d_k$ as destination points.
5. *A set of streams* $\overline{S}$: $\{S_1, ..., S_i\}$.
6. *A routing of* $\overline{S}$ is denoted by $\overline{T}(\overline{S}) = \{T(S_1), ..., T(S_i)\}$.

Given a routing $\overline{T}(\overline{S})$ for $\overline{S}$, we define the following terms :

- *The Loading of an edge e*: $\rho(e) = \sum_{S_i \in \overline{S} \wedge e \in T(S_i)} \varpi_i$, $\forall e \in E$.
- Consider a stream $S \in \overline{S}$. Let $(d, b) \in \{(d_1, b_1), ..., (d_k, b_k)\}$ where d is destination, and b is its associated loss bound, we use $P(s, d)$ to define a path from *s* to *d* in $T(S)$.
  1. *Path loss rate of* $P(s, d)$ is defined as $L(s, d) = \sum_{e \in P(s,d)} L_e(\rho(e))$.
  2. $P(s, d)$ *is valid* for the streams if and only if $L(s, d) \le b$.
- $T(S)$ *is valid* if and only if $P(s, d_i)$ is valid $\forall P(s, d_i) \in T(S)$.
- Consider $\overline{S}_1 \subseteq \overline{S}$, and its corresponding routing $\overline{T}(\overline{S}_1) \subseteq \overline{T}(\overline{S})$, $\overline{T}(\overline{S}_1)$ *is valid* for $\overline{S}$ if and only if $T(S_i)$ is valid $\forall S_i \in \overline{S}_1$.

Now, we are ready to define the BLR problem.

The *BLR problem* is defined below.

Given:

1. *G*=(*V,E*),

2. $L_e(\varpi)$, $\forall e \in E$, where ϖ : traffic load on e,
3. $\overline{S}_a$: the set of accepted streams,
4. $\overline{T}(\overline{S}_a)$: a valid routing for $\overline{S}_a$.

Suppose $\overline{S}_n = \{S_1, ..., S_j\}$ is the set of the new arrival streams.

Note:
If $\overline{S}_a = \varnothing \Rightarrow$ clean network. If $\overline{S}_a \neq \varnothing \Rightarrow$ dirty network.
If $\overline{S}_n = \varnothing \Rightarrow$ clean network. If $\overline{S}_n \neq \varnothing \Rightarrow$ dirty network.

Problem I: Does there exist polynomial time algorithm to find a valid routing $\overline{T}(\overline{S}_n)$ such that the routing $\overline{T}(\overline{S}_n) \cup \overline{T}(\overline{S}_a)$ for $\overline{S}_n \cup \overline{S}_a$ is valid, given that $\overline{T}(\overline{S}_a)$ is valid for $\overline{S}_a$?

Problem II: Find the valid routing $\overline{T}(\overline{S}_n)$ such that the routing $\overline{T}(\overline{S}_n) \cup \overline{T}(\overline{S}_a)$ for $\overline{S}_n \cup \overline{S}_a$ is valid.

Example 1

Consider a network specified by a simple, undirected graph $G=(V,E)$ where $V=\{v_1, v_2, v_3, v_4, v_5, v_6\}$, as shown in Figure 2.

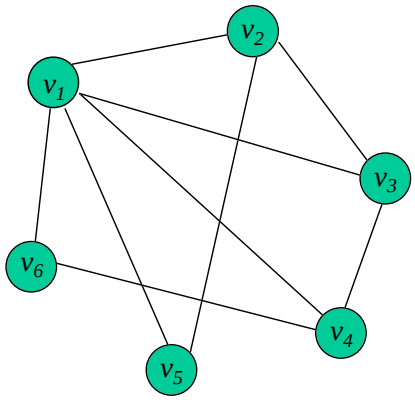

Figure 2. The network model in Example 1.

We assume the loss rate functions of the links are all the same and is depicted in Figure 3.

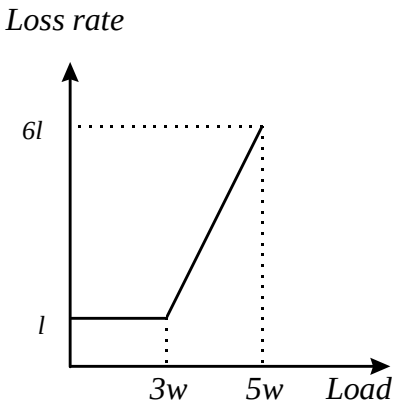

Figure 3. The loss rate function associated with edge in G.

Suppose there are two existing streams: $\overline{S}_a = \{S_1, S_2\}$, where $S_1 = \{v_3, w, (v_2, 3l), (v_6, 4l)\}$ and $S_2 = \{v_4, 2w, (v_2, 3l), (v_5, 5l)\}$ with valid routing $\overline{T}(\overline{S}) = \{T(S_1),\ T(S_2)\}$ as depicted in Figure 4.

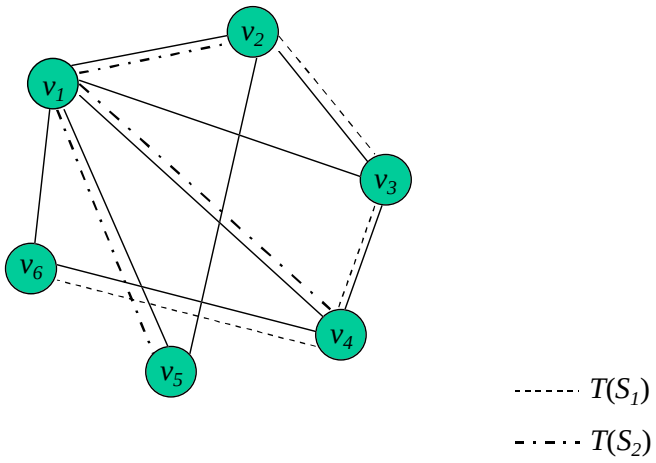

Figure 4. The valid routings of S_1 and S_2.

Consider the new arrival streams $\overline{S}_n = \{S_3, S_4\}$, where $S_3 = \{v_6, w, (v_3, 4l), (v_2, 3l)\}$ and $S_4 = \{v_5, 3w, (v_3, 3l)\}$.

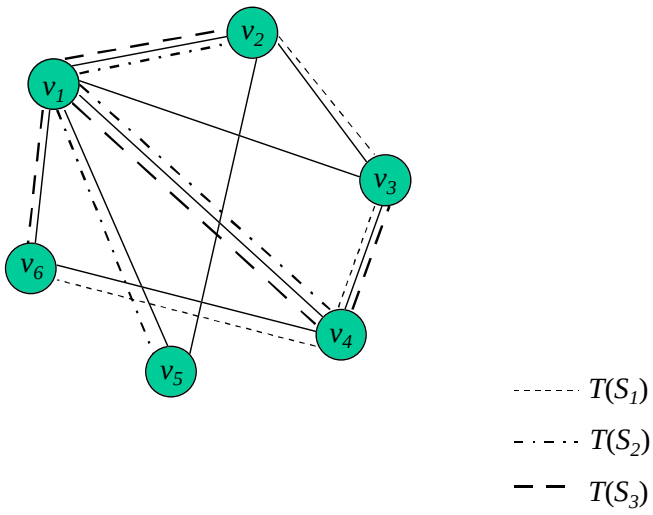

Figure 5. The valid routings of S1, S2 and S3.

For Stream $S_3 = \{v_6, w, (v_3, 4l), (v_2, 3l)\}$, we consider the routing T(S_3) as shown Fig 5.

Notice that the loss rate of
path($v_6 \to v_1 \to v_4 \to v_3$)=$L(w)+L(w+2w)+L(w+w)=3l \leqq 4l$; and
the loss rate of Path($v_6 \to v_1 \to v_2$)=$L(w)+L(w+2w)=2l \leqq 3l$.
Thus T(S_3) is a valid routing!!

For Stream $S_4 = \{v_5, 3w, (v_3, 3l)\}$ however, we can't find any valid routing for T(S_4) !!

In the previous example, it takes three hops for Path($v_6 \to v_1 \to v_4 \to v_3$) in T($S_3$) to connect the source to the destination. In many telephony applications, however, it is often a practical implementation to consider the point-to-point connection with no more than two hops while constructing a multicast tree. Thus we also consider the *constraint BLR* problem in this paper. A *two-hop multicast tree* $T(S)$ is a Steiner tree in *G* with s as the source, d_1 ,.....,d_k as the destination points, and $|P(s, d_i)| \leq 2$ $\forall$ *i*. The *2-hop BLR problem* (or 2hBLR) is the BLR problem except all valid multicast trees are two-hop multicast trees.

4. *Complexity Analysis of the BLR Problem*

The BLR problems can be classified into a few of sub-problems with four system parameters; the number of the existing streams admitted in the system or $\left| \overline{S}_n \right|$, the size of the arrival streams, the multicast degree of the streams, and the hop constraint in the problem. In this paper, we consider two different cases for each of the four parameters; the admitted streams can either be none (the case of the clean network) or some (the case of the dirty network). The new arrival stream can either be one (the single arrival) or many (the batch arrival). The streams can be point-to-point communication (*M=1*) or multicasting (*M>=2*), and finally, the valid multicast tree for each admitted stream is with 2-hop constraint or not. This classification results in sixteen different sub-problems as shown in Figure 6.

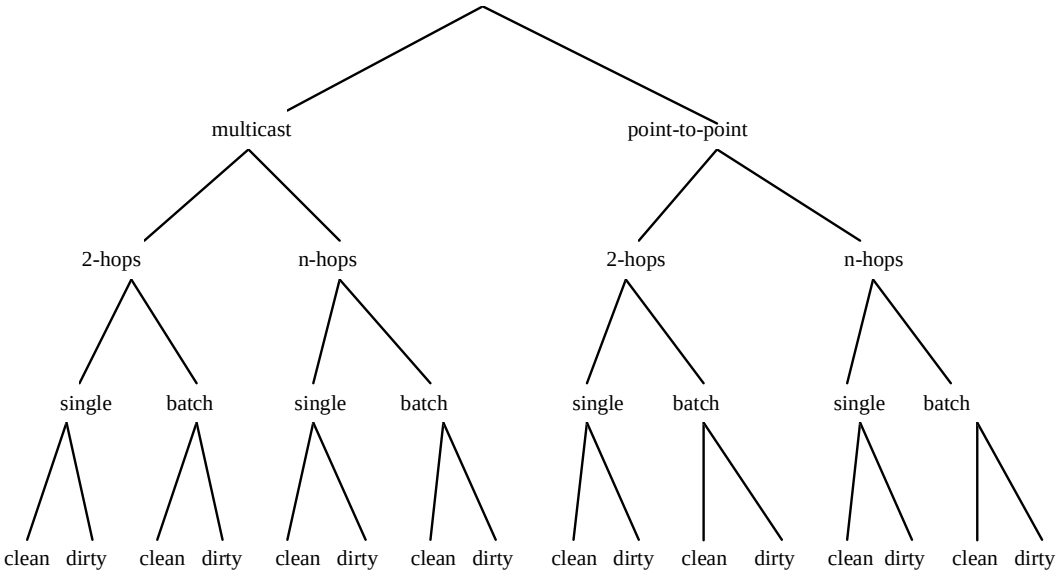

Figure 6. Four parameters classify 16 different sub-problems of the BLR problem.

In this section, we analyze the complexity of the BLR problems. Table 2 summarizes the results of the complexity analysis. For those problems that we have proven NP-complete, the proofs are included in the appendixes. (See Appendix A-E.) For those problems with solutions of linear time complexity are left to the reader as exercises. Finally, For those problems indicated by question marks are still open.

	$\lvert\overline{S}_n\rvert = 1,\ k = 1$ (Single task, point-to-point)	$\lvert\overline{S}_n\rvert > 1,\ k = 1$ (Batch, point-to-point)
$\lvert\overline{S}_a\rvert = 0,\ d = 2$ (Clean network, 2-hop restriction)	Exhausted search $O(V)$	?
$\lvert\overline{S}_a\rvert \neq 0,\ d = 2$ (Dirty network, 2-hop restriction)	Exhausted search $O(V^2)$	NP-complete [1]
$\lvert\overline{S}_a\rvert = 0,\ d > 2$ (Clean network, n hops)	Shortest-path $O(V^2)$	NP-complete [2]
$\lvert\overline{S}_a\rvert \neq 0,\ d > 2$ (Dirty network, n hops)	?	NP-complete

Table 2.a The complexity of the BLR problems.

	$\lvert\overline{S}_n\rvert = 1,\ k \neq 1$ (Single task, multicast)	$\lvert\overline{S}_n\rvert > 1,\ k \neq 1$ (Batch, multicast)
$\lvert\overline{S}_a\rvert = 0,\ d = 2$ (Clean network, 2-hop restriction)	Shortest-path $O(V^2)$	?
$\lvert\overline{S}_a\rvert \neq 0,\ d = 2$ (Dirty network, 2-hop restriction)	$O(V^k)$ All possibilities if k is constant	? if k is constant
	NP-complete [3] otherwise	NP-complete otherwise
$\lvert\overline{S}_a\rvert = 0,\ d > 2$ (Clean network, n hops)	Shortest-path $O(V^2)$	NP-complete [4]
		? if $\lvert\overline{S}_n\rvert = c,\ c$: constant
$\lvert\overline{S}_a\rvert \neq 0,\ d > 2$ (Dirty network, n hops)	NP-complete [5]	NP-complete

Table 2.b The complexity of the BLR problems.

---

[1] See Appendix A.

[2] See Appendix C.

[3] See Appendix B.

[4] See Appendix D.

[5] See Appendix E.

REFERENCES

1. J. Moy, "*Multicast Routing Extensions for OSPF*", *Communication of ACM*, 37(8):61-66, 1994.
2. S.E. Deerimg and D.R. Cheriton, "*Multicast Routing in Datagram Internetworks and Extended LANs*", *IEEE transaction on Computers*, 8(2):85-110, 1990.
3. E. W. Dijkstra, "*A Note on Two Problems in Connexion with Graphs", Numberische Mathemtik*, 1:269-271, 1959.
4. P. Winter, "*Steiner Problem in Networks: A Survey*", *Networks*, 17:129-167, 1987.
5. T. S. Yum and M. S. Chen, "*Multicast Source Routing in Packet-Switched Networks*", *IEEE Transactions on Communication*, 42(2/3/4):1212-1215, 1994.
6. B. K. Kadaba and J. M. Jaffe, "*Routing to Multiple Destinations in Computer Networks*" *IEEE Transactions on Communications*. Com-31(3):343-352, 1983.
7. S. C. Liew, "*Multicast Routing in 3-Stage Clos ATM Switching Networks*", *IEEE Transaction on Communication*, 42(2/3/4):1380-1390, 1994.
8. A. Ciro, J. Noronha and A. T. Fouad,"*Optimum Routing of Multicast streams*", *in Proc. of INFOCOM*, pp. 856-864, 1994.
9. L. Kou, G. Markowsky, and L. Berman, "*A Fast Algorithm for Steiner Trees", Acta Informatica*, vol. 15, pp. 141-145, 1981.
10. V.P. Kompella, J.C. Pasquale, and G.C. Polyzos, "*Multicast Routing for Multimedia Communication", IEEE/ACM transaction on Networking*, vol. 1, no. 3, pp. 286-292, June 1993.
11. Q. Zhu, M. Parsa, and J.J. Garcia-Luna-Aceves, "*A Source-Based Algorithm for Delay-Constrained Minimum-Cost Multicasting", in Proc. of IEEE INFOCOM 95*, 3d.2.1, pp. 377-385, 1995.
12. J. Kadirire, "*Minimizing Packet Copies in Multicast Routing by Exploiting Geographic Spread", ACM SIGCOMM Communication Review*, vol. 24, no.3, pp. 47-62, July 1994.
13. M. Doar and I. Leslie, "*How Bad is Naive Multicast Routing?", in Proc. of IEEE INFOCOM*, San Francisco, CA, pp. 82-89, Apr. 1993.
14. F. Bauer and A. Varma, "*Degree-Constrained Multicasting in Point-to-Point Networks", in Proc. of IEEE INFOCOM 95*, 3d.1.1, pp. 369-376, 1995.
15. J. Kadirire and G. Knight, "*Comparison of Dynamic Multicast Routing Algorithms for Wide-Area Packet Switched (Asynchronous Transfer Mode) Networks", in Proc. of IEEE INFOCOM 95*, 2c.3.1, pp. 212-219, 1995.
16. H. Takahashi and A. Matsuyama. "*An Approximate Solution for the Steiner Problem in Graphs", Math. Japonica*, vol. 24, no. 6, pp. 573-577, 1980.
17. J. Kruskal. "*On the Shortest Spanning Subtree of a Graph and the Traveling*

Salesman Problem", Proc. Amer. Math. Soc., vol. 7, pp. 48-50, 1956.

18. R. Wong. "*A Dual Ascent Approach for Steiner Tree Problems on a Directed Graph", Mathematical Programming*, vol. 28, pp. 271-287, 1984.
19. S. Voss. "*Steiner's Problem in Graphs: Heuristic Methods", Algorithmica*, vol. 7, no. 2-3, pp. 333-335, 1992.
20. B.M. Waxman, "*Routing of Multipoint Connections" IEEE Journal On Selected Areas In Communications*, vol. 6, no. 9, pp. 1617-1622, December 1988.

Appendix A
Dirty network, 2-hop restriction, batch, point-to-point

We want to reduce 3 SAT problem to BLR 2-hop restriction problem.

3 SAT Problem(known as NP-complete)**:**
Instance: Collection $C=\{C_1, C_2,....., C_m\}$ of clauses on a finite set U of literals such that $|C_i| = 3$ for $1 \le i \le m$. (All clauses have exactly three literals per clause.)
Question: Is there a truth assignment for U that satisfies all the clauses in C?

proof:

We define a polynomial-time reduction from 3 SAT problem to BLR 2-hop restriction problem. This reduction transforms a conjuctive normal form Boolean formula *f* to an instance *G* of BLR 2-hop restriction problem. So *f* is satisfiable if and only if *G* can be routed.

Given an instance of 3 SAT Collection $C=\{C_1, C_2,....., C_n\}$ of clauses on a finite set U of literals such that $|C_i| = 3$ for $1 \le i \le n$. We construct an instance of BLR 2-hop as follows: a graph $G = (V, E)$, a source node s ($s \in V$), and a set of destination nodes $D=\{C_1,C_2,...,C_n\}$($D \subseteq V$).

For any clause node *C*, if it contains *x* or $\bar{x}$, it connects to node *x*. If it contains *y* or $\bar{y}$, it connects to node *y*, and so on. Source node s connects to all literal nodes. Please see the following:

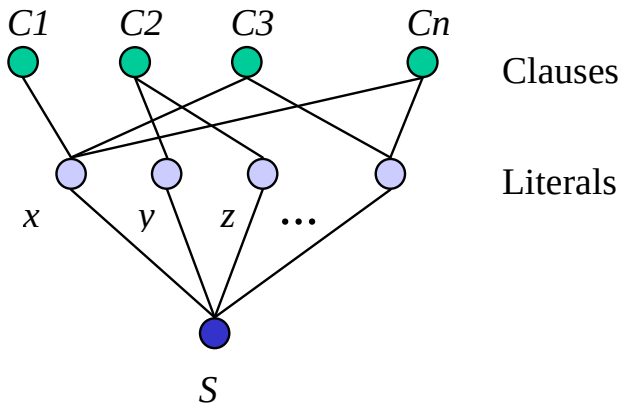

Figure A.1

For every literal *x*:

$L(x)$: a set of clauses with *x* on it.

$L(\overline{x})$: a set of clauses with $\overline{x}$ on it.

For example,

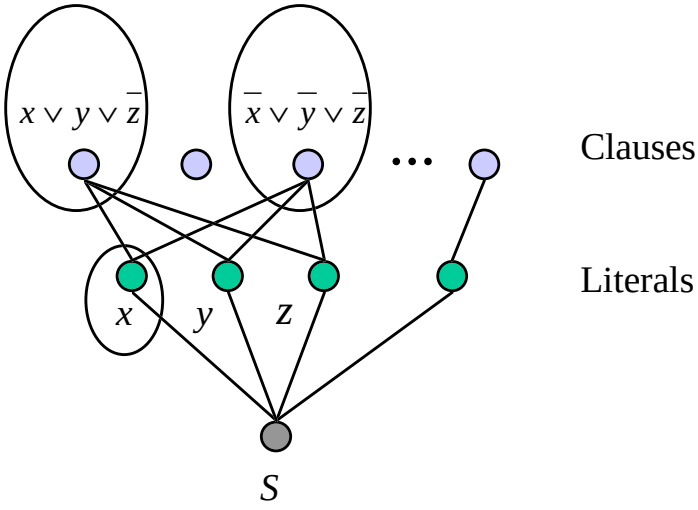

Figure A.2

Let's just consider literal *x* node to all relative nodes (Clauses).

$a_1, \ldots, a_n \in L(x)$

$b_1, \ldots, b_n \in L(\overline{x})$

Path $a_i - x - b_j$ is a routed stream $\forall i \forall j$, where $1 \le i \le n$, $1 \le j \le n$.

Assume there is a loss rate function associated with each edge except edge(s,*x*) in *G*

$L(\rho) = n/$,if $\rho = n$.

$L(\rho) = (n+3)/$,if $\rho = n+1$.

$L(\rho) = 1$ no matter ρ is what value for edge (s, *x*).

*L*oss rate bound for all streams is $(2n+5)/$.

Observe that, every path $a_i - x - b_j$ has load $\rho = n$.

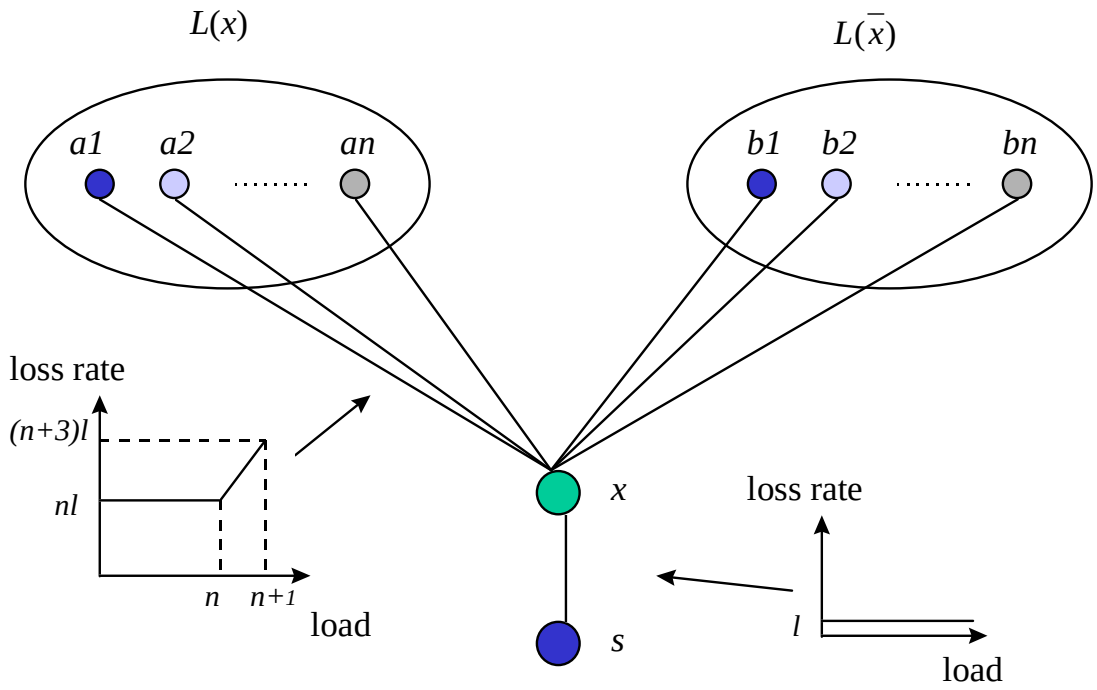

Figure A.3

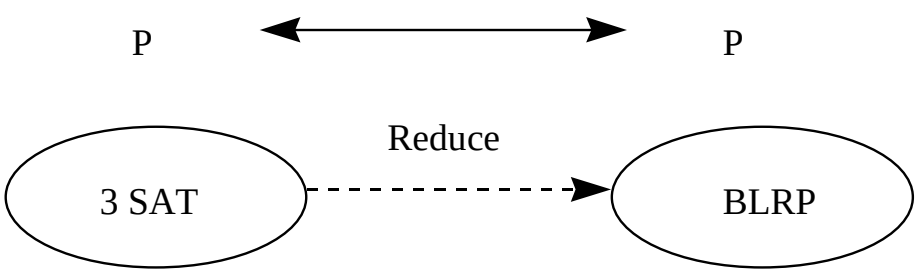

Figure A.4

⇒

If we can get a truth assignment for 3 SAT, we can choose either $L(x)$ or $L(\bar{x})$ from truth assignment. Then we can route BLRP.

⇐

For any admitted stream, the loss rate is *2nl* (*The sum of* a_i*-x* (*nl*) *and x-*b_j (*nl*)). For example, if we want to establish a set of routes from *s* to a_i or b_j, either $L(x)$ or $L(\bar{x})$ group can be chosen for the new route. After choosing $L(x)$ or $L(\bar{x})$, the admitted stream becomes (*2n+3*)*l*, and the new route (*s-x-*a_i or *s-x-*b_j) becomes (*n+4*)*l*. If we choose both $L(x)$ and ($\bar{x}$), the admitted stream will be (*2n+6*)*l*. But the loss rate bound is (*2n+5*)*l*, therefore, we must get a truth assignment (either $L(x)$ or $L(\bar{x})$) to avoid the violation of the loss bound.

Appendix B
Dirty network, 2-hop restriction, single task, multicast BLRP is NP-complete problem

We want to reduce 3 SAT problem to BLR 2-hop restriction problem.

3 SAT Problem(known as NP-complete)**:**
Instance: Collection $C=\{C_1, C_2, ..., C_m\}$ of clauses on a finite set U of literals such that $|C_i| = 3$ for $1 \le i \le m$. (All clauses have exactly three literals per clause.)
Question: Is there a truth assignment for U that satisfies all the clauses in C?

proof:

We define a polynomial-time reduction from 3 SAT problem to BLR 2-hop restriction problem. This reduction transforms a conjuctive normal form Boolean formula *f* to an instance *G* of BLR 2-hop restriction problem. So *f* is satisfiable if and only if *G* can be routed.

Given an instance of 3 SAT Collection $C=\{C_1, C_2, ..., C_n\}$ of clauses on a finite set U of literals such that $|C_i| = 3$ for $1 \le i \le n$. We construct an instance of BLR 2-hop as follows: a graph $G = (V, E)$, a source node s ($s \in V$), and a set of destination nodes $D=\{C_1,C_2,...,C_n\}$ ($D \subseteq V$).

For any clause node *C*, if it contains *x* or $\overline{x}$, it connects to node *x*. If it contains *y* or $\overline{y}$, it connects to node *y*, and so on. Source node *s* connects to all literal nodes. Please see the following:

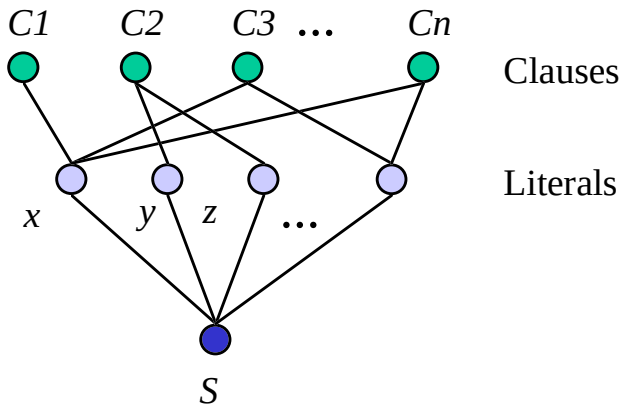

Figure B.1

For every literal *x*:

$L(x)$: a set of clauses with *x* on it.

$L(\overline{x})$: a set of clauses with $\overline{x}$ on it.

For example,

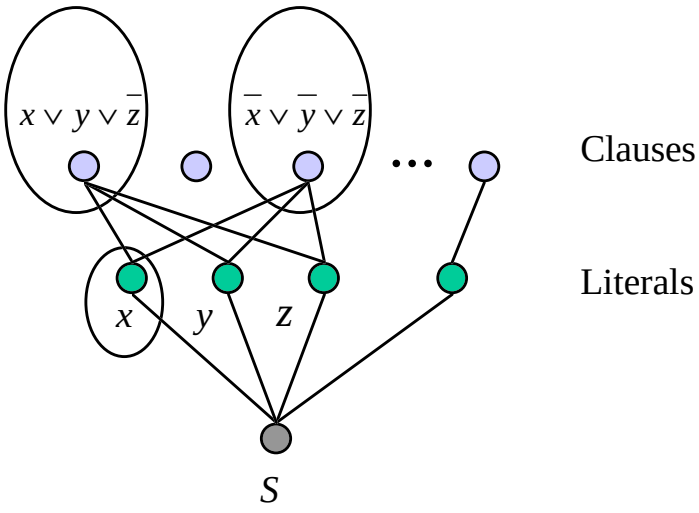

Figure B.2

Let's just consider literal x node to all relative nodes (Clauses).

$a_1, \ldots, a_n \in L(x)$

$b_1, \ldots, b_n \in L(\bar{x})$

Path $a_i - x - b_j$ is a routed stream $\forall i \forall j$, where $1 \le i \le n$, $1 \le j \le n$.

Assume there is a loss rate function associated with each edge except edge(s, x) in G

$L(\rho) = n\,l$,if $\rho = n$.

$L(\rho) = (n+3)\,l$,if $\rho = n+1$.

$L(\rho) = l$ no matter ρ is what value for edge (s, x).

Loss rate bound for all streams is $(2n+5)\,l$.

Observe that, every path $a_i - x - b_j$ has load $\rho = n$.

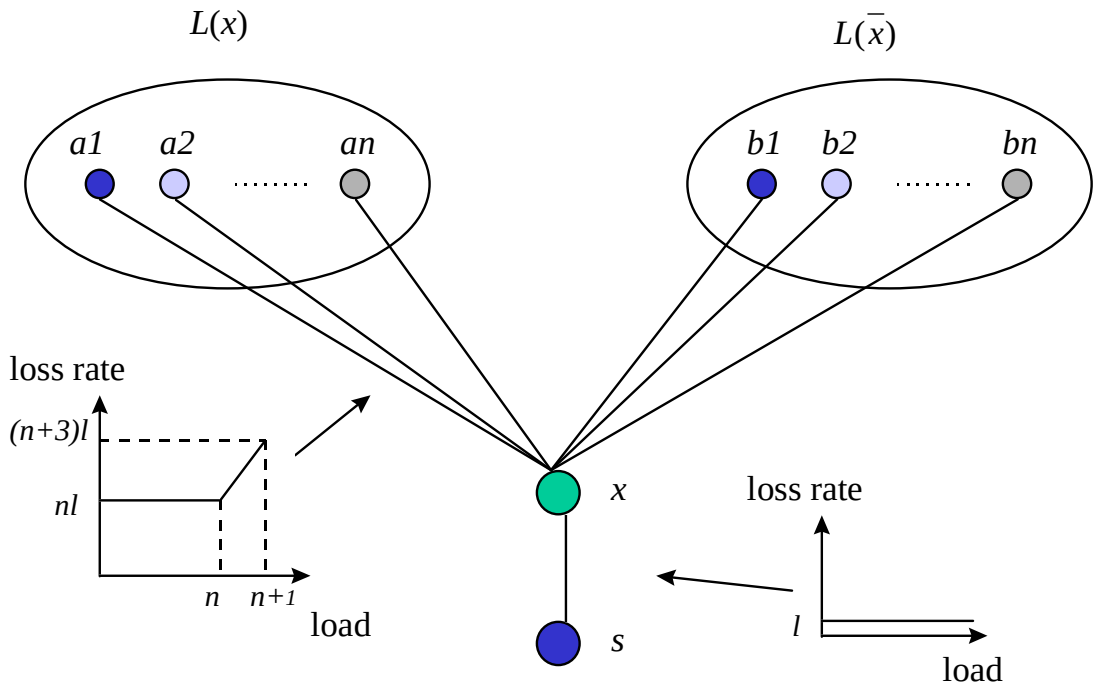

Figure B.3

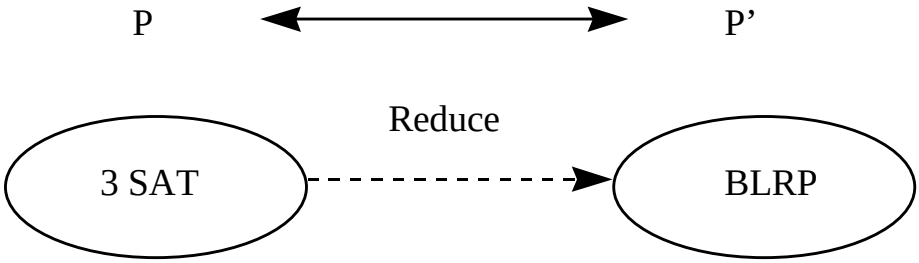

Figure B.4

⇒

If we can get a truth assignment for 3 SAT, we can choose either $L(x)$ or $L(\bar{x})$ from truth assignment. Then we can route BLRP.

⇐

For any admitted stream, the loss rate is $2nl$. If we want to establish a new route from s to a_i or b_j, either $L(x)$ or $L(\bar{x})$ group can be chosen for the new route. After choosing $L(x)$ or $L(\bar{x})$, the admitted stream becomes $(2n+3)l$, and the new route becomes $(n+4)l$. If we choose both $L(x)$ and $L(\bar{x})$, the admitted stream will be $(2n+6)l$. But the loss rate bound is $(2n+5)l$, therefore, we must get a truth assignment (either $L(x)$ or $L(\bar{x})$) to avoid the violation of the loss bound.

Appendix C
Proof for Clean network, n-hop, batch, point-to-point

We want to reduce *Maximum Length-bounded Disjoint Paths Problem* to this problem.

Maximum Length-bounded Disjoint Paths Problem(known as NP-complete)**:**
Instance: Graph $G=(V,E)$, specified vertices s and t, positive integers J, $K \leq |V|$.
Question: Does G contain J or more mutually edge disjoint paths from s to t, none involving more than K edges ($K \geq 5$)?

proof:

We define a polynomial-time reduction from problem to our problem. This reduction transforms graph G to an instance G' of clean network, n-hop, batch, point-to-point problem. So that G is satisfiable if and only if G' can be routed.

Given an instance of G. We construct an instance of clean network, n-hop, batch, point-to-point problem as follows:

A graph G' with source s and destination t, we construct $s_1,.....,s_j$ to connect with s, and $t_1,.....,t_j$ to connect with t. We want to route j streams from s_i to t_i.

Assume there is a loss rate function associated with each edge in G.

$L(\rho) = l$ if $\rho = 1$,

$L(\rho) = 15\,l$ if $\rho = 2$.

Loss rate bound for all streams is $7\,l$.

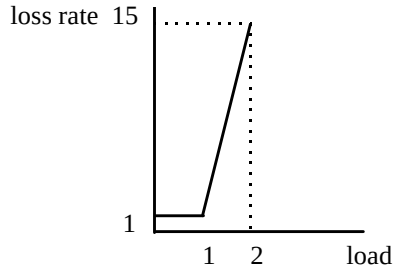

Figure C.1

*Garey,Johnson, "Computer and Intractability" 1979.

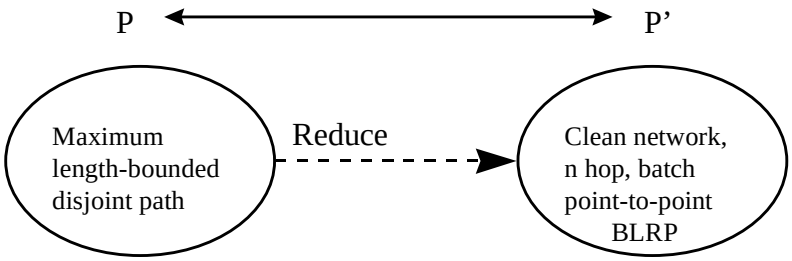

Figure C.2

⇒

If we can solve Maximum Length-bounded Disjoint Paths Problem , then we can route clean network, *n* hop, batch, point-to-point BLRP.

⇐

1. Since the bound is 7 *l* , the path length for every stream in the "black box" can't be more than 5 hops. It can map to term "no more than *K* edges".
2. Any two streams can't use the same link in the "black box". Since if they use the same link, loss rate will be more than 15 *l* . It can map to term "edge-disjoint".

We must route those j streams from s_i to t_i in this situation, then we can find a solution to Maximum Length-bounded Disjoint Paths Problem.

Appendix D
Proof for Clean network, n-hop, batch, multicast

We want to reduce M*aximum Length-bounded Disjoint Paths Problem* to this problem.

Maximum Length-bounded Disjoint Paths Problem:
Instance: Graph G=(V,E), specified vertices s and t, positive integers J, $K \leq |V|$.
Question: Does G contain J or more mutually edge disjoint paths from s to t, none involving more than K edges ($K \geq 5$)?

proof:

We define a polynomial-time reduction from problem to our problem. This reduction transforms graph G to an instance G' of clean network, n-hop, batch, point-to-point problem. So that G is satisfiable if and only if G' can be routed.

Given an instance of G. We construct an instance of clean network, n-hop, batch, point-to-point problem as follows:

A graph G', j streams with the same source s and the same destination t.

Assume there is a loss rate function associated with each edge in G.

$L(\rho) = 1$ if $\rho = 1$,

$L(\rho) = 15\,l$ if $\rho = 2$.

Loss rate bound for all streams is $5\,l$.

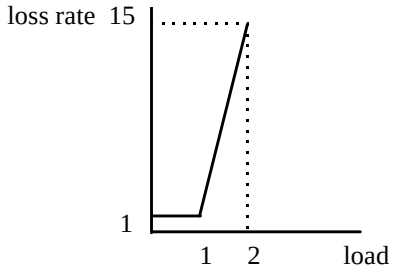

Figure D.1

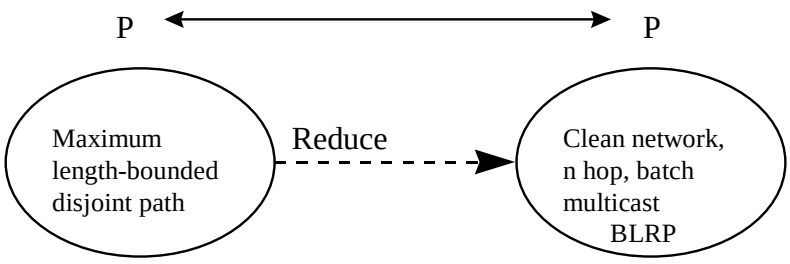

Figure D.2

⇒

If we can solve Maximum Length-bounded Disjoint Paths Problem , then we can route clean network, n hop, batch, multicast BLRP.

⇐

1. Since the bound is $5l$, the path length for every stream in the "black box" can't be more than 5 hops. It can map to term "no more than K edges".
2. Any two streams can't use the same link in the "black box". Since if they use the same link, loss rate will be more than $15l$. It can map to term "edge-disjoint".

We must route those j streams in this situation, then we can find a solution to Maximum Length-bounded Disjoint Paths Problem.

Appendix E

Dirty network, n hop, single task, multicast BLRP is NP-Complete problem

We want to reduce 3 SAT problem to BLRP.

3 SAT Problem(known as NP-complete)**:**

Instance: Collection $C=\{C_1, C_2,....., C_m\}$ of clauses on a finite set U of literals such that $|C_i| = 3$ for $1 \le i \le m$.

Question: Is there a truth assignment for U that satisfies all the clauses in C?

proof:

We define a polynomial-time reduction from 3 SAT problem to BLRP. This reduction transforms a conjuctive normal form Boolean formula *f* to an instance *G* of BLRP. So that *f* is satisfiable if and only if G can be routed.

Given an instance of 3 SAT Collection $C=\{C_1, C_2,....., C_n\}$ of clauses on a finite set U of literals such that $|C_i| = 3$ for $1 \le i \le n$. We construct an instance of BLRP as follows:

a graph $G = (V, E)$, a source node $s(s \in V)$, and a set of destination nodes $D=\{C_1,C_2,...,C_n\}(D \subseteq V)$. $V=\{C_1,C_2,...,C_n, s\} \cup \{ l_i^f , l_i^m , l_i^t$ for all l. i∈ all literals in clauses} (where f means false, t means true and m is mid .)

Source node connects to all l_i^m nodes.

For example, $C_m=(l_i \vee \bar{l}j \vee l_k)$, this node connects to l_i^t , l_j^f , l_k^t. This can be done in polynomial time.

Assume there is a loss rate function associated with each edge in G . $L(\rho)= l$ if ρ is 1, $L(\rho)= 3l$ if ρ is 2 and Loss rate bound is $5.5l$. Moreover, every path has load $\rho=1$. Assume every path $l_i^f \rightarrow l_i^m \rightarrow l_i^t \ \forall i$ has load 1.

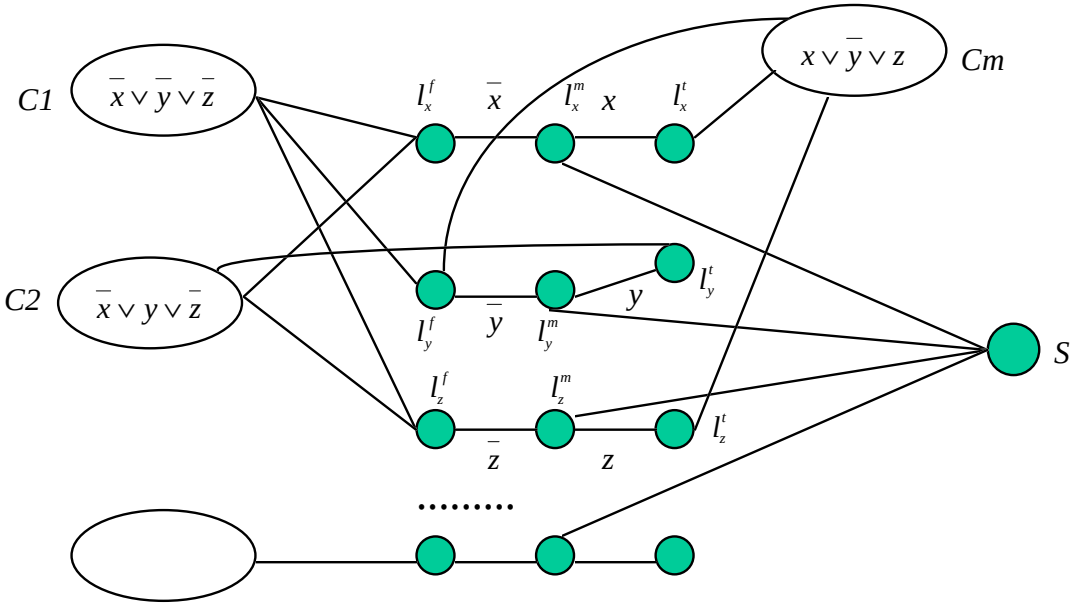

Figure E.1

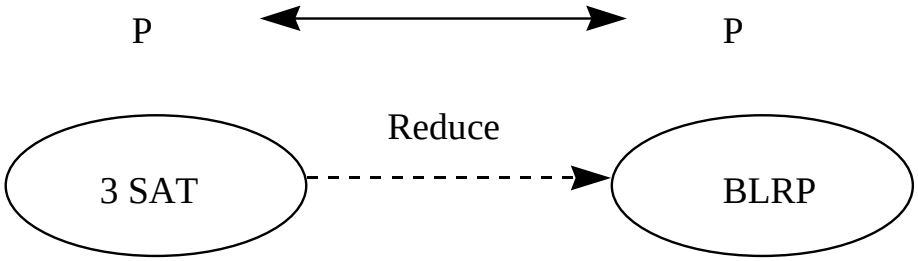

Figure E.2

⇒

If we can get a truth assignment for 3 SAT, we can choose either x or $\bar{x}$ from truth assignment. Then we can route BLRP.

⇐

For any admitted stream, the loss rate is 2l. For example, if we want to establish a new route from s to C_m, either x or $\bar{x}$ can be chosen for the new route. After choosing x or $\bar{x}$, the admitted stream becomes 4l, and the new route becomes 5l. If we choose both, the admitted stream will be 6l. Therefore, we can get a truth assignment.

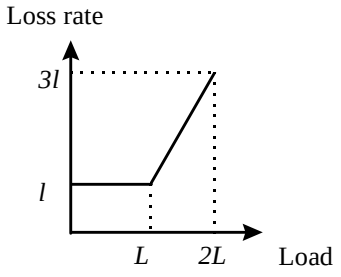

Figure E.3