MAMCRA:

A Constrained-Based Multicast Routing Algorithm

Fernando Kuipers and Piet Van Mieghem

Delft University of Technology,
Information Technology and Systems,
P.O. Box 5031, 2600 GA Delft, The Netherlands.
June 19, 2001

Abstract
Multicast routing algorithms that are capable of providing quality of service (QoS) to its members will play an important role in future communications networks. This paper discusses some fundamental properties of multicast routing subject to multiple QoS requirements. We will show that guaranteeing QoS and optimizing resource utilization are conflicting objectives and require a trade-off. We also present MAMCRA, a Multicast Adaptive Multiple Constraints Routing Algorithm, that guarantees QoS to the multicast members in an efficient, but not always optimal manner.

Key-words: routing, multicast, quality of service (QoS), Steiner, tree, MAMCRA

Submitted to: Computer Communications
Corresponding author: Fernando Kuipers
Delft University of Technology
Information Technology and Systems
P.O. Box 5031, 2600 GA Delft
The Netherlands
F.A.Kuipers@its.tudelft.nl
Tel.: +31-15-2781347

1. Introduction: Multicast Routing

The main problem considered in this paper is that of routing from a single source node to a set of p destination nodes, also called multicast source routing or point-to-multipoint routing. The advances in technology and the fast emerging multimedia applications have provided great impetus for new (real-time) multicast applications. A frequently encountered example of a real-time multicast application is videoconferencing, but many others will emerge in the future. Videoconferencing, as a good representative of the class of real-time multicast applications, requires sufficient bandwidth and, in addition, limits to the maximum delay, jitter and (packet) loss. These requirements, which we consider the same for all members, are referred to as Quality of Service (QoS) constraints. Many multicast applications will not operate properly if QoS cannot be guaranteed. Hence, future multicast algorithms must be capable of satisfying a set of (feasible) QoS-constraints.

A main property of multicast routing is the efficient use of resources (Van Mieghem *et al*., 2001a). Because each of the p destination nodes will receive the same information, unicast (sending the information p times over each shortest path to each individual multicast participant), is inefficient since most likely there will be some overlap among the set of shortest paths. Multicasting as few duplicate packets as possible and only duplicating them if necessary clearly is more efficient. For the case of a single metric, multicast source routing can be implemented by forwarding the packet of a flow or session over the shortest path tree. The more general problem of multipoint-to-multipoint leads to the optimal solution of the minimum Steiner tree problem (Hwang *et al*., 1992).

The main contributions of this paper are ordered as follows. Section 2 will give a formal definition of the main problems of multicast routing with multiple constraints. Since all previous research on multicast routing focuses on finding a multicast tree (e.g. see Salama *et al.*, 1997) and often only considers a fixed number of constraints (e.g. only delay and jitter in Rouskas and Baldine, 1997), we shall discuss multiple-constrained multicast trees in section 3. We will demonstrate that finding a tree subject to multiple QoS requirements is not always possible. Section 4 will briefly overview SAMCRA, our previously proposed unicast QoS-algorithm (Van Mieghem *et al*., 2001b). In section 5 we will extend this algorithm to a multicast QoS-algorithm called MAMCRA, Multicast Adaptive Multiple Constraints Routing Algorithm. MAMCRA finds the set of shortest paths to all destinations and then reduces the consumption of resources without violating the QoS-constraints. Section 6 gives a discussion on multicast routing and poses some suggestions, after which section 7 concludes this paper.

2. Problem Definition

A communication network is modeled as an undirected graph $G(N, E)$, where N is the set of nodes and E is the set of links[1]. Each link is characterized by a link vector $\vec{w}$ consisting of m link metrics w_i, for $i = 1, \ldots, m$. We presume that the full network

[1] With a slight abuse of the notation, we sometimes also denote the cardinality $|S|$ of the set S by S. Thus, the number of nodes $|N|$ in the set N, by N and similar for the number of links E.

topology is known at a certain time interval and regard the topology metrics as frozen. A network topology supporting QoS consists of link vectors with non-negative QoS-metrics as components. The QoS-metric of a path can either be *additive* in which case it is the sum of the QoS-metrics of the links along the path (such as delay, jitter, the logarithm of packet/cell loss, cost of a link, etc.) or it can be the *minimum(maximum)* of the QoS-metrics along the path (typically available bandwidth and policy flags). Min(max) QoS-metrics are treated by omitting all links (and possibly disconnected nodes) that do not satisfy the requested min(max) QoS-constraints. This is called topology filtering. This paper only considers m additive QoS-metrics, which cause more difficulties (as explained below) than min(max) QoS-constraints. The m QoS-constraints L_i, for $i = 1, \ldots, m$ are represented by the constraint vector $\vec{L}$. Since all participants receive the same multicast application emitted by the source, the constraint vector $\vec{L}$ is the confining vector for all multicast members of the group.

A multicast sub-graph $M(\{s, D\}, H) \subseteq G(V, E)$ has $p < N$ multicast destination nodes (multicast group members or participants) represented by the set $D = \{d_1, \ldots, d_p\}$. Each of these destination nodes is connected to the source node s, by the links in $H \subseteq E$. As will be exemplified below, M is best regarded as a set of paths from s to d_j, j=1,..p, which use the links in H.

Definition of a cycle: A cycle is a path in a graph, which starts and ends at the same node and includes other nodes at most once.

Definition of dominance: Let $\vec{a}$ and $\vec{b}$ be two different vectors, each consisting of m components. $\vec{a}$ dominates $\vec{b}$ if $a_i \le b_i$, for $i = 1, \ldots, m$, with an inequality sign for at least one i.

Definition of non-dominance (Henig, 1985): A vector $\vec{a}$ is called non-dominated if there does not exist a vector $\vec{b}$ that dominates $\vec{a}$.

Problem I (Multiple Constrained Multicast (MCM)):
Given s and D, find M({s, D}, H) such that for each path P(s,d_j) from s to $d_j \in D$, j=1,…p:

$$w_i(P) \le L_i, \qquad \text{for } i = 1, \ldots, m$$

where $\vec{w}(P)$ is the vector sum of the links that constitute P:

$$w_i(P) = \sum_{e \in P} w_i(e), \qquad \text{for } i = 1, \ldots, m.$$

Note that if for a certain $d_j \in D$ no feasible path exists, d_j should be removed from D.

Define $\vec{w}(M) = \sum_{1 \le l \le |H|} \vec{w}(e_l \in H)$ and $l(M) = \|\vec{w}(M)\|$ as the length (or vector norm) of $\vec{w}(M)$. The length $l(M)$ can be any function $f(\vec{w}(M))$ on the weight vector of M that returns a real number, provided that $f(.)$ is a vector norm (see 4.2). Throughout this paper

we consider $l(M) = \max_{i=1,\ldots,m}\left(\frac{w_i(M)}{L_i}\right)$, which has been motivated in our study of unicast QoS routing (De Neve and Van Mieghem, 2000).

Problem II (Multiple Parameter Steiner Tree (MPST)):
For s, D given, find M({s, D}, H) for which l(M) is minimum.

Problem III (Multiple Constrained Minimum Weight Multicast (MCMWM)):
For s, D given, find M({s, D}, H) such that for each path P(s,d_j) from s to $d_j \in D$, j=1,…,p:

$w_i(P) \le L_i$, *for* $i = 1, \ldots, m$

and

l(M) is minimum

Problem III is a combination of problems I and II.

Section 3 will further discuss these three problems: solving the first problem results in satisfying the QoS requirements, the second minimizes the total resource consumption and the third optimizes the resources subject to the QoS requirements. Clearly, the last problem is the most desirable objective for QoS multicast routing.

3. Properties of Multicast QoS Routing

First, all above defined problems are shown to be NP-complete. Subsequently, the MCM and MPST problem are demonstrated to lead to non-compatible solutions.

Theorem I: *MCM, MPST and MCMWM are NP-complete*

Proof:

MCMWM is a combination of MCM with MPST. Therefore by proving that MCM is NP-complete, we will also have proved that MCMWM is NP-complete.

Let us first consider MCM. For $D = \{d\}$ this problem reduces to unicast QoS-routing, which is proved to be NP-complete for $m \ge 2$ additive parameters (Garey and Johnson, 1979; Wang and Crowcroft, 1996).

For m = 1, MPST reduces to the minimum Steiner tree problem, which is known to be NP-complete (Karp, 1972). QED.

In this paper we focus on solving the MCMWM problem that can be considered the hardest of the 3 problems.

Property 1: *The MPST does not necessarily obey the constraints for all multicast members, even if there exist feasible paths towards these members.*

Proof: It suffices to prove property 1 by providing an example. Consider the topology in Figure 1.

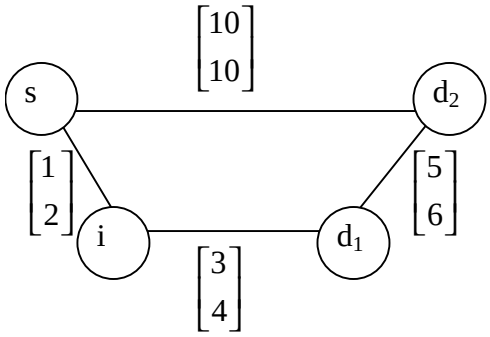

Figure 1. Example topology.

Here *s* is the source node, *i* is some intermediate node, and d_1 and d_2 are the two destination nodes participating in the multicast session. The MPST connecting *s*, d_1, d_2 is the path *s*-*i*-d_1-d_2 with a total weight vector of (1, 2) + (3, 4) + (5, 6) = (9, 12). If the constraints were (13, 13), then this would be the best solution to the MCMWM problem, since all the QoS constraints are met with a minimum consumption of resources. However, if the constraints are more stringent, say (11, 11) then the path from *s* to d_2 exceeds these constraints. In this case the multicast tree [(*s*-*i*-d_1), (*s*-d_2)] obeys the requested constraints. This tree has a weight vector of (1, 2) + (3, 4) + (10, 10) = (14, 16) and is the 2^{nd}-shortest MPST. QED.

We have proved that the MPST, although optimal in terms of resource utilization, does not always satisfy the constraints. Since in the example topology of Figure 1 the 2^{nd}-shortest MPST was the best solution, this suggests that considering *k*-shortest MPST will lead to the optimal solution for MCMWM. We will demonstrate that this is not always the case. We will illustrate that in order to guarantee QoS, we need to abandon the concept of trees in multiple dimensions. *Only for a single metric, the MCMWM solution is a tree.* If the solution to MCMWM would always be a tree, then *k*-shortest MPST would give the exact solution.

Property 2: *The graph (or topology) solution of the MCM and MCMWM is not necessarily a tree.*

Proof: Again, this property is demonstrated via an example.

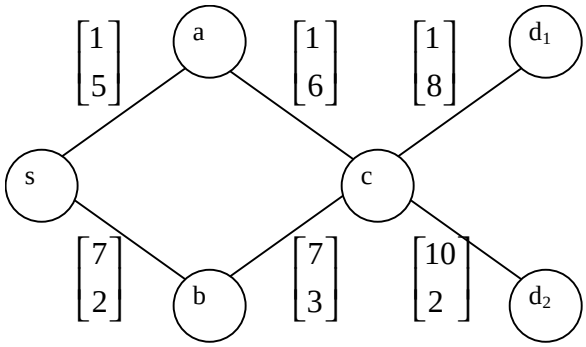

Figure 2. Example topology.

The MPST for the topology in Figure 2 consists of the links *s-a, a-c, c-d_1* and *c-d_2* and has a total weight (vector) of (1, 5) + (1, 6) + (1, 8) +(10, 2) = (13, 21). The path weight vector for the path from *s* to *d_1* is (3, 19) and for *s* to *d_2* is (12, 13).

The 2nd-shortest MPST consists of links *s-b, b-c, c-d_1, c-d_2* and has weight vector (7, 2) + (7, 3) + (1, 8) + (10, 2) = (25, 15). The path weight vector is (15, 13) between *s* and *d_1* and (24, 7) between *s* and *d_2*.

In this example, no other non-dominated trees connecting *s, d_1, d_2* exist, i.e. none of the components of the weight vectors of the remaining (two) trees connecting *s, d_1, d_2* are smaller than those of the given weight vectors. Therefore, if the above-mentioned trees do not satisfy a certain QoS, then no tree can satisfy this QoS.
If the constraints are (16, 16) then no tree can provide the requested QoS. The only way to obey these constraints is by means of two paths: *s-b-c-d_1* and *s-a-c-d_2*. In that case the multicast sub-graph, *M({s, D}, H)*, is not a tree. QED.

Property 3: *The solution to MPST is always a tree.*

Proof:

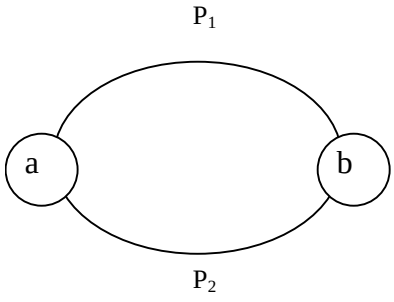

Figure 3. A cycle formed by the paths ("branches") P_1 and P_2.

If the solution *M* to MPST is not a tree, then it must contain a cycle as depicted in Figure 3. The length of this solution *M* equals *l(M)* and according to problem definition II, *l(M)* must be minimum, i.e.:

$$l(M) \leq l(M') \quad \forall M'$$

Without loss of generality we assume that *M* only contains the cycle depicted in Figure 3. Further we define *M' = M\{P₂}*, i.e. *M'* is a tree.
Since all weights are positive, we have:

$$l(M) = l(\sum_{e \in H \setminus (P_1, P_2)} \vec{w}(e) + \vec{w}(P_1) + \vec{w}(P_2))$$
$$\geq l(\sum_{e \in H \setminus (P_1, P_2)} \vec{w}(e) + \vec{w}(P_1)) = l(M')$$

This leads to a contradiction, because according to problem definition II $l(M) \leq l(M')$. Therefore, the sub-graph *M* (containing the cycle) cannot be a solution to MPST. QED.

4. Unicast QoS Routing

First we will give an overview of SAMCRA, Self-Adaptive Multiple Constraints Routing Algorithm, our previously proposed unicast QoS routing algorithm (Van Mieghem *et al.*, 2001b), to give some insight into multiple-constrained routing and because we will use SAMCRA as a basis for our multicast QoS routing algorithm in section 5. Next we will discuss the definition for length.

4.1 SAMCRA

SAMCRA returns the path between a given source and destination subject to the (end-to-end) constraints L_i on each QoS measure ($1 \leq i \leq m$). SAMCRA is based on three fundamental concepts: a non-linear measure for the path length, the *k*-shortest path approach and the principle of non-dominated paths.
All *m* additive QoS measures are considered as equally important. Each link is specified by a *m*-dimensional weight vector $[w_1, w_2, ..., w_m]$. The path vector $\vec{w}(P) = \{w_1(P), w_2(P), ..., w_m(P)\}$ is the vector sum of the link weights along this path, thus, $w_i(P) = \Sigma_{k \to l \in P} w_i(k \to l)$ for each *i*. The path length is a vector norm and given by

$$l(P) = max_{1 \leq i \leq m}(w_i(P)/L_i) \quad (1)$$

This definition (1) obeys the criteria for 'length' or 'distance' in vector algebra (see 4.2) and is motivated by the geometry of the constraints surface in *m*-dimensional space. We claim that (1) is extendable to any transformation $f(\vec{w}(P))$ of the *m*-vector components to a positive real number, provided that $\vec{w}(P)$ lies within the constraints and obeys the criteria for length. For example, some QoS measures may be regarded as more important than others, min-max measures can also be considered. Of course, the difficulty lies in the motivation of *f*. Subsection 4.2 is devoted to this discussion. In the sequel, we limit the focus to the definition (1) for which we refer to (De Neve and Van Mieghem, 2000) for a detailed discussion. An important corollary of a non-linear length function such as (1) is that *the subsections of shortest paths in multiple dimensions are not necessarily shortest paths.* This corollary suggests to consider in the computation more paths than only the shortest one, leading us naturally to the *k*-shortest path (Chong, 1995) approach (i.e. we consider the shortest path, the 2nd shortest, etc. up to the k^{th} shortest path). Finally, the multi-dimensional character of QoS routing invites the use of state space reduction, which has been implemented via the concept of non-dominated paths (Henig, 1985). The number of shortest paths stored, during the computation of SAMCRA, in

each nodal queue is represented by *k*. There always exists a finite value of *k*, which is upper bounded by

$$k_{\max} = \min\left(\frac{\prod_{i=1}^{m} L_i}{\max_j (L_j)}, \lfloor e(N-2)! \rfloor \right) \quad (2)$$

The first argument of the min-operator in (2) refers to the number of relevant path vectors within the constraints surface. The second argument, where $\lfloor x \rfloor$ denotes the largest integer equal to or smaller than *x* and *e = 2.718...*, is an attainable upper bound for the number of paths between 2 nodes in any graph with *N* nodes (Van Mieghem, 1998, 2001c).
SAMCRA is an exact algorithm in the sense that it always returns the shortest path according to the length definition.

Complexity:
The worst-case complexity of SAMCRA is O($kNlog(kN) + k^2mE$), with $k = k_{max}$ given by (2).
Simulations on millions of random graphs reveal that $k << k_{max}$ (Kuipers and Van Mieghem, 2001). (For *m* = 1, SAMCRA turns into Dijkstra's algorithm).

4.2 On The Definition of Length in QoS Routing

The definition *l(.)* of the length of a vector $\vec{p}$ must satisfy the following criteria (Golub and Van Loan, 1983; Royden, 1988):
(1) $l(\vec{p}) > 0$ for all non-zero vectors and $l(\vec{p}) = 0$ only if $\vec{p}$ contains one or more zeros.
(2) for all vectors $\vec{p}$ and $\vec{u}$ holds the triangle inequality

$$l(\vec{p} + \vec{u}) \le l(\vec{p}) + l(\vec{u}) \quad (3)$$

If $\vec{p}$ and $\vec{u}$ are non-negative vectors (i.e. all vector components are non-negative), we have

$$l(\vec{p} + \vec{u}) \ge l(\vec{p}) \quad (4)$$

because the length of a non-negative vector cannot decrease if a non-negative vector is added.
For example, if $\vec{p} = [1, 3, 5, 1, 9]$, $\vec{u} = [4, 5, 2, 1, 0]$ and $L = [10, 1, 10, 10, 1]$, then, using (1), $\vec{p} + \vec{u} = [5, 8, 7, 2, 9]$, $l(\vec{p}) = 9$ and relation (4) gives $l(\vec{p} + \vec{u}) = l(\vec{p}) = 9$.

Depending on the constrained optimization problem, we can use SAMCRA with different length functions, provided they obey the criteria for length. This subsection is devoted to the motivation of suitable length functions for QoS routing, by giving three length functions for some problems encountered in QoS routing. The three problems considered are the Multiple-Constrained Path (MCP) problem, the Delay-Constrained Least-Cost (DCLC) path problem and the Hop-Constrained Maximum Bandwidth (HCMB) problem.

Multiple-Constrained Path (MCP):

Given a graph G(N, E) with m metrics per link and a constraint vector $\vec{L}$*, find a path P from source s to destination d that obeys all constraints of the constraint vector.*

SAMCRA was designed to solve this problem, moreover with the length function (1), SAMCRA not only finds a feasible path (if one exists), but the path is optimized for all metrics. SAMCRA may however be stopped if a feasible path is found, instead of continuing the search via (1) for the constraints-optimized path (the path that lies as far as possible from all the constraints).

Delay-Constrained Least-Cost (DCLC) path:
Given a graph G(N, E) where each link is characterized by a delay and (monetary) cost, a source node s and a destination node d. Given a delay-constraint D, the problem is to find a path P within the delay-constraint for which the cost is minimum.

A suitable length function for this problem is:

$$l(P) = \begin{cases} \dfrac{c(P)}{C}, & \text{if } d(P) \le D \\ \infty, & \text{else} \end{cases} \qquad (5)$$

where $C = N \max_{1 \le l \le E}(c(e_l))$ is a cost-constraint that each cycle-free path can obey, *c(P)* is the cost of path *P* and *d(P)* is the delay of *P*. The length function (5) only optimizes for one metric, the cost (price). The price is often considered the most important metric to minimize.
Although it is tempting to call this length function linear since it only optimizes a single metric, the constraint(s) make (5) non-linear (the function is linear only within the constraints surface). Therefore, again, the subsections of shortest paths are not necessarily shortest paths. E.g. see Figure 4 with *D* = 15: The least-cost path from *s* to *d* is *s*-*b*-*d* with cost 2 and delay 20. This path exceeds the delay constraint and therefore the length is set to ∞. The path *s*-*a*-*b*-*d* does obey the delay-constraint and therefore is the shortest path from *s* to *d*, but its sub-path from *s* to *b* is not the shortest path to *b*.
Guo and Matta (1999) have successfully used this approach to solve the DCLC problem.

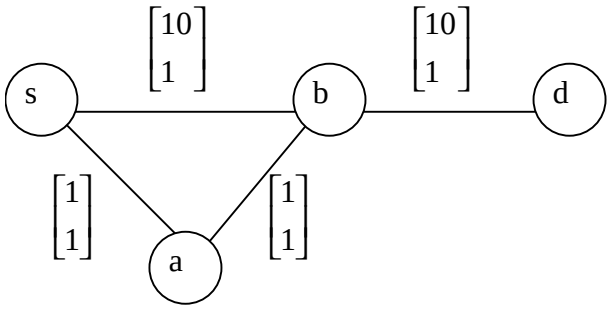

Figure 4. Example topology. Top metric corresponds to delay and bottom metric to cost.

Hop-Constrained Maximum-Bandwidth (HCMB) problem:

Given a graph G(N, E), where each link has a specified capacity (bandwidth). Given a source s and a destination d, find a path P with no more than X hops that has maximum capacity.

A suitable length function for this problem is:

$$l(P) = \begin{cases} \dfrac{1}{BW(P)}, & if\ h(P) \le X \\ \infty, & else \end{cases} \qquad (6)$$

where *BW(P)* is the capacity of path *P* in minimum units (e.g. Mb/s) and *h(P)* is the number of hops taken by *P*. The length function (6) is very similar to (5), but is given to illustrate that min/max parameters can also be incorporated. Note that besides a different length function, this problem requires us to make a small but simple change to SAMCRA's code, because *BW(P)* is not the sum of the capacities of the links on *P*, but the minimum link capacity on *P*. A discussion of the HCMB problem can be found in (Apostolopoulos *et al.*, 1999; Guérin and Orda, 2000).

We have shown that, provided a suitable length is chosen, SAMCRA applies to numerous constraint/optimization problems. MAMCRA, the multicast version of SAMCRA, can therefore be used for all types of constrained-based (optimization) problems.

5. MAMCRA

This section presents the algorithm MAMCRA, the Multicast Adaptive Multiple Constraints Routing Algorithm. The solution provided by MAMCRA solves problem I exactly and approximates problem III in the sense that is does not always find the multicast sub-graph *M* with minimum weight (= minimum resource consumption). Although MAMCRA was not designed to solve problem II, it may also be considered a heuristic to this problem. The quality of MAMCRA with respect to problem II is topic for further study.

MAMCRA operates as follows:

A. First the set *S* of shortest paths from *s* to all *p* multicast members is calculated.
B. Optimize *M*, i.e. reduce *l(M)*, without violating the constraints

Step A in the basic MAMCRA operation is readily obtained since it only requires a small modification of SAMCRA. SAMCRA's stop condition (line 8 in the meta-code below) is altered, so that it only stops if all destinations (within the constraints) have been reached. Since S(M)AMCRA operates in a Dijkstra-like manner, during the computation of the set of shortest paths to the multicast members, we may find shortest paths to other destinations. One may choose to also include these paths in the set *S*. Then, if one of these destinations decides to join the multicast session, we already have a compliant path. When removing the overlap of paths, the set *S* may lead to a tree, but this tree may not be optimal in terms of resource consumption. E.g. if we again consider the example topology of Figure 1, with the constraints (13, 13), the set *S* consists of the paths *s-i-*d_1

and s-d_2, which form a tree. The tree s-i-d_1-d_2, however, also obeys the constraints and is more efficient in terms of resource consumption.
Step A is easily completed and provides us with a solution to MCM. We just need to remove the overlap in the set S, so that duplicate packets are only generated when necessary. The elimination of overlap (including the check on min/max constraints) will be addressed below and in section 6.

Step B requires some more effort. Some additional terminology is needed.

We define the concatenation of two paths P and Q by PQ, i.e. PQ is the path generated by appending path Q to path P. Note that $\vec{w}(PQ) = \vec{w}(P) + \vec{w}(Q)$.
With $\overset{d}{\leq}$ we refer to non-dominance, i.e. $\vec{w}(P) \overset{d}{\leq} \vec{w}(Q)$ means that $\vec{w}(Q)$ is dominated by $\vec{w}(P)$.

Consider two paths $P_1(s,d_1)$ and $P_2(s,d_2)$ that form a cycle, i.e. both paths have two nodes in common. The first node in common is the source node and the other is node $x \in N\backslash\{s, d_1, d_2\}$. If the two paths have more than two nodes in common, we have a concatenation of cycles with x the (common) node that is most hops away from s.

Property 4:
If

$$\vec{w}(P_2(s,d_2)) - \vec{w}(P_2(s,x)) + \vec{w}(P_1(s,x)) \overset{d}{\leq} \vec{L}$$

then $P_2(s,d_2)$ may be rerouted to $P_1(s,x)P_2(x,d_2)$ without violating the constraints.

Proof:

Let $P_{old} = P_2(s,d_2)$ and $P_{new} = P_1(s,x)P_2(x,d_2)$, with $\vec{w}(P_{old}) \overset{d}{\leq} \vec{L}$
If

$$\vec{w}(P_{old}) - \vec{w}(P_2(s,x)) + \vec{w}(P_1(s,x)) \overset{d}{\leq} \vec{L}$$

and the fact that

$$\vec{w}(P_{old}) - \vec{w}(P_2(s,x)) = \vec{w}(P_2(x,d_2))$$

Then

$$\vec{w}(P_2(x,d_2)) + \vec{w}(P_1(s,x)) = \vec{w}(P_{new}) \overset{d}{\leq} \vec{L}$$

QED.

Property 4 tells us that by removing cycles, we are optimizing the total weight vector, since:

$$\vec{w}(P_1(s,d_1)) + \vec{w}(P_2(x,d_2)) \overset{d}{\leq} \vec{w}(P_1(s,d_1)) + \vec{w}(P_2(s,d_2))$$

For example consider Figure 2, where $P_1(s,d_1) = s\text{-}b\text{-}c\text{-}d_1$ and $P_2(s,d_2) = s\text{-}a\text{-}c\text{-}d_2$. The total weight vector of these two paths is (27, 26). Assuming that property 4 holds, we can reroute $P_2(s,d_2)$ to $P_1(s,x)P_2(x,d_2)$. The total weight vector has now reduced to (25, 15).

Property 5:
Given a path P(s,d) within the constraints that uses the sub-path P(s,a), then P(s,a) also lies within the constraints (but is not necessarily the shortest path from s to intermediate node a).

Property 5 follows from the basic property of a non-linear length in multiple dimensions provided in (Van Mieghem *et al.*, 2001b), namely that sub-paths of shortest paths in multiple dimensions are not necessarily shortest paths.

Meta-code:
The meta-code of MAMCRA is divided into two parts A and B. A gives the adjusted code of SAMCRA (lines 8, 9) and B gives the code for optimizing the resource utilization.

A.

```
1. counter = 0 for all nodes
2. endvalue = 1.0
3. path(s[1]) = NULL and length(s[1]) = 0
4. put s[1] in queue
5. while(queue ≠ empty)
6.    EXTRACT_MIN(queue) -> u[i]
7.    u[i] = marked gray
8.    if all members have been extracted
9.          STOP and return the set of shortest paths (S)
10.   else
11.         for each v ∈ adjacency_list(u)
12.               if(v ≠ previous node of u[i])
13.                     PATH = path(u[i]) + (u,v)
14.                     LENGTH = length(PATH)
15.                     check all non-black paths at v and PATH for
                        dominancy, endvalue -> mark obsolete paths black
16.                     if(LENGTH ≤ endvalue and non-dominated)
17.                           if(paths are not black)
18.                                 counter(v) = counter(v) + 1
19.                                 j = counter(v)
20.                                 path(v[j]) = PATH
21.                                 length(v[j]) = LENGTH
22.                                 put v[j] in queue
23.                           else
24.                                 replace a black path with PATH
```

Lines 1 to 4 are initializations, line 1 initializes the counter for each node to zero. This counter keeps track of the number of entries in the queues. To start the algorithm with the source node, this node is inserted into the queue (line 4). The EXTRACT_MIN function (see Cormen *et al*., 2000) in line 6 selects the minimum path length in the queues and returns the associated node *u* with its entry number *i*, which is the *i*-th path stored in the queue at node *u*. The extracted node is marked gray in line 7. The first time that a destination node is extracted, we store this node and its entry number, so that we can back

trace this shortest path at the end of the algorithm. If the extracted node *u* equals the last destination (member) that was not yet extracted, all the shortest paths satisfying the constraints are returned; else the scanning procedure is invoked (lines 11 and 12). Similar as in Dijkstra's algorithm, the neighbors of the extracted node are examined. Line 13 describes how the path up to node *u* is extended towards the neighboring node *v*. Line 15 checks for path dominance as explained in (De Neve and Van Mieghem, 2000). If an old unmarked path is dominated by the new entry PATH, it is marked black. A node marked black has become obsolete and may be replaced by a new path. If the path is within the constraints and not dominated, it can be added to the queue. Lines 17 to 24 describe how path(length)s for a node can be added. If there are black paths in the queue, a black path may be replaced by the new path, else the path must be added as a new entry.

B.

```
1. If S ≠ ∅
      add the path with most members (d_j) to M. (If there are more
      maximum member paths available, choose the one with smallest
      length).
   else
      return M
2. If the newly added path forms a cycle in M, try to optimize M by
   means of property 4.
3. Check if the new path does not violate the min/max constraints
4. Remove from S all paths to nodes that are already visited by M
   (property 5).
5. go to 1.
```

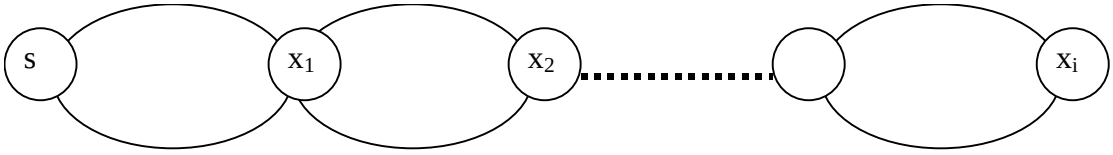

Figure 5. Concatenation of cycles.

In part B of MAMCRA's meta-code, by sequentially adding and optimizing paths the set of shortest paths *S* found in part A is lowered in order to obtain a multicast sub-graph *M* that uses as few links from *S* as possible. In step 1 if the set S is not empty, this means that at least one member is not part of *M* yet. If there are multiple paths in *S* left, we choose the one that traverses most members. In case there are multiple paths with the largest number of members, the path with smallest length is chosen. The chosen path is added to *M*. In step 2, the newly added path may form multiple cycles (< *N*) as depicted in Figure 5. In that case we first try to optimize for all cycles, by considering them as being one large cycle[2] $s->x_i->s$, where *i* equals the number of cycles. If this is not possible, we repeat the procedure without examining the last cycle, i.e. $s->x_{i-1}->s$. When a cycle cannot be removed/optimized, this means that some overlap may be introduced that cannot be removed, i.e. *M* is not a tree and therefore some link(s) may see duplicate packets. When considering the min/max constraint bandwidth, this means that the link has to be able to provide more bandwidth than the bandwidth consumption of the source, i.e. the capacity of the link must be equal or larger than *r* times the bandwidth constraint,

[2] Since the weight vector of a concatenation of cycles equals the sum of the individual weight vectors of the cycles, this assumption is justified, with only a slight abuse of the definition of a cycle.

where *r* equals the number of replicated packets on the link. Therefore, *if a tree cannot be formed, an additional check on the min/max constraints is required*. This check is made in step 3. This procedure is repeated until S is empty, which means that M contains all feasible members.

Complexity:
The worst-case complexity is O($kNlog(kN) + k^2mE$), with $k = k_{max}$ given by (2) for part A and O(Np^2) for part B.

Example of MAMCRA:

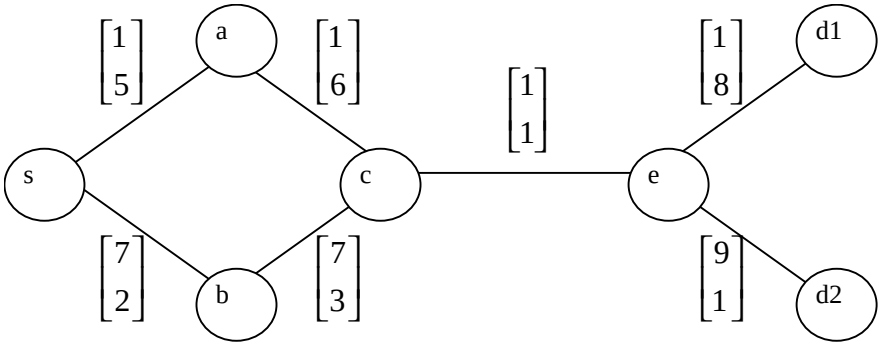

Figure 6. Example topology

	s	a	b	c	e	d_1	d_2
0	(0,0)						
1		s(1,5)	s(7,2)				
2				a(2,11)			
3				b(14,5)			
4					c(3,12)		
5						e(4,20)	e(12,13)
6							Shortest path to d_2 found
7					c(15,6)		
8						e(16,14)	e(24,7)
9							Shortest path to d_1 found
							STOP

Table 1. Execution steps of MAMCRA part A

Table 1 shows the steps taken by MAMCRA in part A, given the topology of Figure 5. During initialization, the source node is set to (0, 0). In the next step *s* scans its neighbors and finds paths to *a* and *b*, with path vectors respectively (1, 5) and (7, 2). We also keep track of the previous node. If we use the length function (1), vector (1, 5) is the smallest entry and therefore *a* is extracted next and the scanning procedure is repeated. This process of extracting and scanning is continued until both destinations have been extracted once. By back-tracing the paths from d_1, d_2 to *s* we receive the set $S = \{(s\text{-}b\text{-}c\text{-}e\text{-}d_1), (s\text{-}a\text{-}c\text{-}e\text{-}d_2)\}$.

So far, the constraints have not yet been taken into account. (This is an exception, used solely to simplify this example and because the constraints are identical this does not alter the true operation. Normally, part A also includes the constraints).
Now we optimize *S* (part B) to gain *M*, using two scenarios. First the constraints are (20, 20):

1. We add (s-a-c-e-d_2) to *M*, because this path is shortest in length where all paths have the same amount of members (only one).
2. No cycle was formed, so we add (s-b-c-e-d_1) to *M*
3. (s-b-c-e-d_1) creates a cycle (s-a-c-b-s) in *M*. When property 4 is applied, we see that $\vec{w}(s\text{-}b\text{-}c\text{-}e\text{-}d_1) - \vec{w}(s\text{-}b\text{-}c) + \vec{w}(s\text{-}a\text{-}c) = (16, 14) - (14, 5) + (2, 11) = (4, 20) \overset{d}{\leq} (20, 20)$. We therefore reroute ($s$-$b$-$c$-$e$-$d_1$) to ($s$-$a$-$c$-$e$-$d_1$).

The result is $M = \{(s\text{-}a\text{-}c\text{-}e\text{-}d_1), (s\text{-}a\text{-}c\text{-}e\text{-}d_2)\}$, which can be written as a tree $M = \{(s\text{-}a\text{-}c\text{-}e),(e\text{-}d_1),(e\text{-}d_2)\}$ if we remove the overlap.

If the constraints are (16, 16), we cannot optimize *S* and therefore $M = S = \{(s\text{-}a\text{-}c\text{-}e\text{-}d_2), (s\text{-}b\text{-}c\text{-}e\text{-}d_1)\}$.

Part B constructs the multicast sub-graph by sequentially adding paths. This approach allows MAMCRA to add new members to an existing "tree". Part A first calculates the paths to the new members, after which part B sequentially and efficiently adds them to the existing sub-graph. However, re-computing the entire multicast sub-graph is hardly more intensive and may therefore be preferred. The choice of how to add/remove members is part of the QoS multicast protocol (see 6.3).

6. Discussion

6.1 Tuning MAMCRA

MAMCRA gives an efficient but not always optimal solution to MCMWM. It is possible to further optimize MAMCRA by considering not only the shortest paths to d_j (j=1,…,p), but by storing all $k_{requested}$-shortest paths (within the constraints) from s to d_j in *S*. The cost of optimizing MAMCRA in this way lies in complexity (running time). Each node now has a queue-size $k_{requested} \leq k \leq k_{max}$. Although this approach does not alter the worst-case complexity of part A (after all, MAMCRA already works with *k*-shortest paths, the only difference now is that we examine at least $k_{requested}$ shortest paths instead of as few as possible), it will have an effect on the running time, which will increase proportional to k^2 (see MAMCRA's complexity). The worst-case complexity of part B becomes $O(kNp^2)$. However, the larger we choose *k*, the higher the probability that we find the exact solution to MCMWM. Therefore *k* is considered to be a tuning parameter. Note that since part B is a heuristic, this approach will never be able to guarantee that the exact solution to MCMWM is always found.

6.2 QoS Negotiation

We have indicated that guaranteeing QoS and optimizing resource utilization are two conflicting interests. Depending on the wishes of the client (multicast member), a trade-

off can be made between QoS and resource utilization. This trade-off will be based on monetary cost, since guaranteeing a high level of QoS will inflict a large consumption of resources, which has to be paid for. It is not likely that all members are willing to pay the same price. It would therefore be beneficial if some sort of negotiation between QoS and price could take place. For instance, MAMCRA (in step A) computes the set of shortest paths based on the maximum allowable constraints ($\overrightarrow{L_{\max}}$). In the optimization phase, the different wishes ($\overrightarrow{L_{d_j}} \overset{d}{\leq} \overrightarrow{L_{\max}}$) of the members can be taken into account. For instance if a certain level of QoS (within the $\overrightarrow{L_{\max}}$ constraints) has a high price and another slightly worse level of QoS (also within the $\overrightarrow{L_{\max}}$ constraints) has a significant lower price, the client may negotiate his requested level of QoS. As argued in section 4, SAMCRA only examines the solutions within the constraints and hence we can apply any length-function, provided it obeys the criteria for length (see 4.2). Because price is often considered the most important parameter to minimize, we can take $l(P) = w_i(P)$, where w_i corresponds to the price metric (see 4.2).

6.3 QoS Multicast Protocol

In the first scenario of our example (Figure 6) both paths use *s*-*a*-*c*-*e* and we therefore need to send a packet only once over this sub-path, after which it is duplicated at node *e* and sent to d_1 and d_2. In the second scenario the paths have an overlap on *c*-*e*. Since the packets at the source are duplicated and forwarded to *a* and *b*, duplicate packets arrive at node *c*. These duplicate packets have traveled a different path towards *c* and have different weights. E.g. one packet may arrive later at *c* than its duplicate counterpart, but its price/loss/jitter may be less. Since the paths from *c* to the destinations also have different weights, we must maintain both packets at link *c*-*e*. Only allowing one packet on *c*-*e* would results in a violation of the constraints of one of the destinations. Thus we can only remove an overlap on *s* to some node *i*. As argued earlier, if we have an overlap, we should check if the min/max constraints are still guaranteed or perhaps renegotiate the constraints (6.2). The task of efficiently forwarding/replicating packets is part of the multicast protocol in use and not of MAMCRA. Several traditional multicast protocols exist, like DVMRP (RFC 1074), MOSPF (RFC 1583) and PIM (RFC 2362). These protocols were designed for best-effort traffic. The protocols in (Carlberg and Crowcroft, 1997; Faloutos *et al*., 1998 and Chen *et al*., 2000) are better suited for delivering QoS. However these and other protocols only consider multicast *trees*. MAMCRA therefore either requires a new or modified multicast protocol. This protocol has to cope with different dynamics, e.g. network dynamics or the joining/leaving of multicast members. It must ensure stability of the multicast "tree", but it must also (efficiently) guarantee QoS; which are conflicting targets. Since providing QoS is the goal, also some type of resource reservation is desirable.

It has been a goal of this paper to address the difficulties in providing guaranteed multicast QoS. We have seen that the constraints, imposed by guaranteed QoS, introduce numerous difficulties mainly related to the possible overlap (caused by the absence of a tree) and hence an objective of a network provider should be to always strive towards a multicast tree.

6.4 Multicast in an Active Network

Inspired by Connectionless Multicast (CLM) (Bovie et al., 2000) we touch upon diffserv multicast and its exact active counterpart. In CLM, the packet header carries the IP addresses of all the multicast members. Each router determines the next hop for each destination and constructs a new header for every distinct hop. The new header only contains destinations for which the next hop is on the shortest path. In conformance with unicast diffserv, we can extend CLM, such that each packet belongs to a certain Class of Service (CoS) and each router has a routing table for each CoS. We have proved (Van Mieghem *et al*., 2001b) that destination-based QoS routing can only be guaranteed in an active network. If we store the history of an active packet in its header, then for each packet arriving at a router, MAMCRA is used to compute the best forwarding/replication strategy. The best use for such an active strategy is in highly dynamic (e.g. wireless) environments, since we do not need (to recalculate) routing tables. However, we do need to have an accurate view of the network.

7. Conclusions

We have proposed an algorithm to find an efficient multicast graph in a network that obeys a set of QoS-constraints. We have shown that a multicast tree may not always guarantee the requested QoS-constraints, while multiple unicast QoS sessions can. This property enhances the complexity of constrained multicast routing (besides the proven NP-completeness), since we have to maintain a set of paths/trees and we need to check if no min/max constraints are violated (merely topology filtering may be insufficient). A trade-off between efficient use of resources and QoS has to be made, which resulted in the proposed algorithm MAMCRA. MAMCRA computes the set *S* of shortest paths from source *s* to all other nodes, and then reduces this set to an efficient set of multicast routes, without compromising the requested level of QoS. It was shown that it is desirable to always construct (or strive for) a multicast tree, either by fine-tuning MAMCRA or by renegotiating the constraints.

Acknowledgements

We would like to thank Hans De Neve from Alcatel CRC Antwerp, for his useful comments.

References

Apostolopoulos, G., D. Williams, S. Kamat, R. Guérin, A. Orda and T. Przygienda, 1999, "*QoS Routing Mechanisms and OSPF extensions*", RFC 2676, August.

Carlberg, K. and J. Crowcroft, 1997, "*Building Shared Trees Using a One-to-Many Joining Mechanism*", Computer Communication Review, pp. 5-11, January.

Chen, S., K. Nahrstedt and Y. Shavitt, 2000, "*A QoS-Aware Multicast Routing Protocol*", INFOCOM 2000.

Chong E. I., S. Maddila, S. Morley, 1995, "*On Finding Single-Source Single-Destination k Shortest Paths*", J. Computing and Information, special issue ICCI'95, pp.40-47.

Cormen T. H, C. E. Leiserson and R. L. Rivest, 2000, "*Introduction to Algorithms*", MIT Press, Boston.

De Neve, H. and P. Van Mieghem, 2000, "*TAMCRA: A Tunable Accuracy Multiple Constraints Routing Algorithm*", Computer Communications, vol. 23, pp. 667-679.

Faloutos, M., A. Banerjea and R. Pankaj, 1998, "*QoSMIC: Quality of Service sensitive Multicast Internet protoCol*", SIGCOMM '98, September.

Garey M. R. and D. S. Johnson, 1979, "*Computers and Intractability, A Guide to the Theory of NP-Completeness"*, Freeman, San Francisco.

Golub, G. H. and C. F. Van Loan, 1983, "*Matrix Computations*", North Oxford Academic, first edition, Oxford.

Guérin, R. and A. Orda, 2000, "*Computing Shortest Paths for Any Number of Hops*", submitted for publication, http://www.seas.upenn.edu:8080/~guerin/ .

Guo, L. and I. Matta, 1999, "*Search Space Reduction in QoS Routing*", in Proceedings of the 19th International Conference on Distributed Computing Systems, Austin, Texas, June.

Henig, M. I., 1985, "*The shortest path problem with two objective functions*", European J. of Operational Research 25, pp.281-291.

Hwang, F.K., D.S. Richards and P. Winter, 1992, "*The Steiner Tree Problem*", North-Holland.

Karp, R., 1972, "*Reducibility among Combinatorial Problems*", Complexity of Computer Communications (R. Miller and J. Thatcher, eds.), Plenum Press, pp. 85-103.

Kuipers, F. and P. Van Mieghem, 2001, "*QoS routing: Average Complexity and Hopcount in m dimensions*", to appear in proceedings of QofIS2001 (2nd International Workshop on Quality of future Internet Services).

Bovie, R., N. Feldman, Y. Imai, W. Livens, D. Ooms and O. Paridaens, 2000, "*Explicit Multicast (Xcast) Basic Specifications*", <draft-ooms-xcast-basic-spec-00.txt>.

Rouskas, G.N. and I. Baldine, 1997, "*Multicast Routing with End-to-End Delay and Delay Variation Constraints*", IEEE JSAC, vol. 15, no. 3, April.

Royden, H. L., 1988, "*Real Analysis*", Macmillan Publishing Company, 3rd edition, New York.

Salama, H.F., D.S. Reeves and Y. Viniotis, 1997, "*Evaluation of Multicast Routing Algorithms for Real-Time Communication on High-Speed Networks*", IEEE JSAC, February.

Van Mieghem, P., 1998, "A lower bound for the end-to-end delay in networks: Application to voice over IP", IEEE Globecom'98, Nov. 8-12, Sydney (Australia), pp. 2508-2513.

Van Mieghem, P., G. Hooghiemstra and R. van der Hofstad, 2001a, "*On the Efficiency of Multicast*", to appear in IEEE Transactions on Networking.

Van Mieghem, P., H. De Neve and F.A. Kuipers, 2001b, "*Hop-by-Hop quality of Service Routing*", to appear in Computer Networks.

Van Mieghem, P., 2001c, "*Paths in the simple Random Graph and the Waxman Graph*", to appear in Probability in the Engineering and Informational Sciences (PEIS).

Wang, Z. and J. Crowcroft, 1996, "*QoS Routing for supporting Multimedia Applications*", IEEE J. Selected Areas in Communications, 14(7):1188-1234, September.