Hop-by-hop quality of service routing

Van Mieghem, PFA; de Neve, H; Kuipers, FA

DOI
https://doi.org/10.1016/S1389-1286(01)00222-5

Publication date
2001

Document Version
Accepted author manuscript

Published in
Computer Networks

HOP-BY-HOP QUALITY OF SERVICE ROUTING

Piet Van Mieghem[(1)], Hans De Neve[(2)] and Fernando Kuipers[(1)]

(1) Delft University of Technology, Information Technology and Systems, P.O. Box 5031, 2600 GA Delft, The Netherlands.

(2) Alcatel Corporate Research, Francis Wellesplein 1, B-2018 Antwerp, Belgium.

ABSTRACT

Based on SAMCRA, an improved and exact version of our QoS routing algorithm TAMCRA, we have investigated QoS routing in a hop-by-hop manner because it forms the basis of IP networking as e.g. in OSPF. In particular, we studied 'hop-by-hop destination based only' (HbHDBO) QoS routing that ignores the source and previous path history (as in current IP routing). We demonstrate that an exact QoS algorithm assures the avoidance of routing loops in this HbHDBO setting. However, despite the use of an exact QoS routing algorithm as SAMCRA, the exact solution cannot be guaranteed with HbHDBO routing. Fortunately, large simulation results on various sizes of random graphs show that the overall quality of the HbHDBO QoS routing is remarkably good. Finally, we show that, by using active networking as opposed to current IP routing, exact QoS routing in a hop-by-hop way can be guaranteed.

Key-words: routing, quality of service, NP-completeness, (T)SAMCRA

SUBMITTED TO: Computer Networks

April 20, 2000

Corresponding author: Piet Van Mieghem

Delft University of Technology

Information Technology and Systems

P.O. Box 5031, 2600 GA Delft,

The Netherlands

P.VanMieghem@its.tudelft.nl

tel.: (31)-15-278 23 97

1. Introduction: HbHDBO Routing.

The main motivation of this article is the question whether current OSPF can be extended to multiple constraints QoS routing. Our study may be seen as a complement to that of Apostolopoulos *et al.* in RFC 2676. We assume the existence of a QoS supporting routing protocol of the link state family that is capable of offering each node in the network a consistent view of the topology. Obtaining a 'consistent' view of the topology is a complex problem that has currently attracted a lot of interest (see also PNNI). The difficulty of obtaining complete topology information has lead to researches on QoS routing in networks with inaccurate information (Guérin and Orda, 1999). Apostolopoulos *et al.* (1999) broadly discuss dynamic QoS routing aspects, while Chen and Nahrstedt (1999) focus on dynamic QoS routing in wireless media (also called ad hoc networks). The dynamic updating of the topology link metrics due to coupling with the resource consumption in the nodes of the network is regarded here as beyond the scope and is treated elsewhere (Van Mieghem and De Neve, 1998). Presuming the full knowledge of the network topology at a certain time interval during which we regard the topology metrics as frozen, we focus here on *algorithmic* aspects of (static) QoS routing using TAMCRA (De Neve and Van Mieghem, 1998, 2000) or its exact modification SAMCRA. T(S)AMCRA will be briefly reviewed in sec. 2.

A network topology supporting QoS consists of link metrics vectors with non-negative QoS measures as components. The QoS measure of a path can either be *additive* in which case it is the sum of the QoS measures 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 measures along the path (typically, available bandwidth and policy flags). Min(max) QoS measures are treated by omitting all links (and possibly disconnected nodes) which do not satisfy the requested min(max) QoS measure. We call this topology filtering. Additive QoS measures are expected to cause more difficulties: the multiple constraints problem (MCP), defined as finding a path subject to more than one additive constraint, is known to be NP-complete (Garey and Johnson, 1979, Wang and Crowcroft, 1996) and hence, considered as intractable for large networks.

Section 3 deals with the feasibility of 'hop-by-hop destination based only' (HbHDBO) QoS routing. Although QoS routing naturally asks for on-demand, end-to-end and explicit routing (Sales and Van Mieghem, 1998), we focus on the MCP for routing in a connectionless environment as shown in Figure 1. The study is motivated by the fact that, first of all, 'diffserv' routing is a missing and desirable network architectural building block and further, that, apart from policy (or inter-domain) routing, all routing protocols in the Internet operate in a hop-by-hop

mode. Thus, it directly impacts the question whether OSPF is extendable to provide QoS with a sufficient level of 'guarantee hardness'.

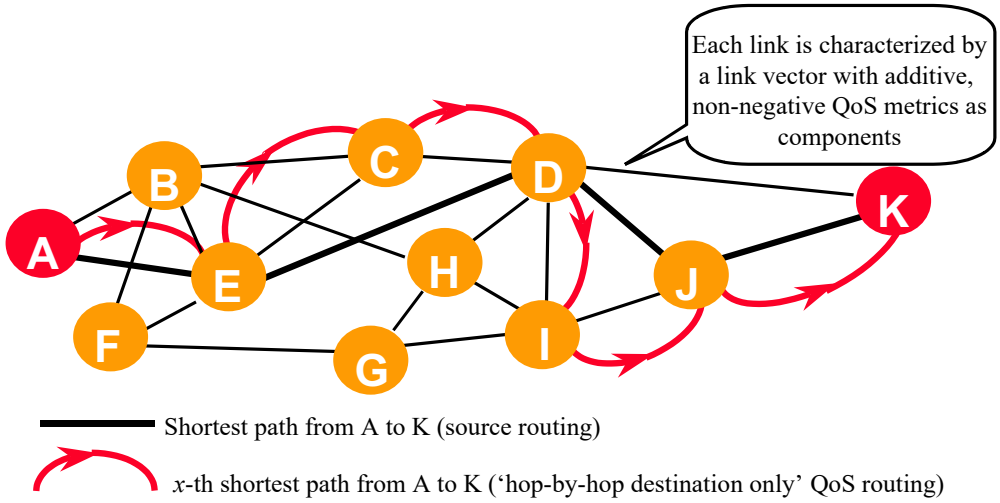

Figure 1 Comparison of exact end-to-end QoS routing (via source routing) and HbHDBO QoS routing.

In that section 3, we show that (a) HbHDBO QoS routing with an exact QoS algorithm such as SAMCRA assures loop-freeness, that (b) even with an exact algorithm and as opposed to single parameter routing (as Dijkstra), no end-to-end constraints can be guaranteed. Immediately, the question about the quality of HbHDBO QoS routing arises. Section 4 presents simulations[1] of the performance of HbHDBO QoS routing in random graphs with uniformly distributed link weights. For that class of networks, we find that the differences with respect to the exact result are remarkably small. Finally, section 5 demonstrates that, by using active networking as opposed to current IP routing, exact QoS routing in a hop-by-hop way can be guaranteed.

2. QoS Routing: T(S)AMCRA

Here we briefly review our QoS routing algorithm, a Tunable Accuracy Multiple Constraints Routing Algorithm (TAMCRA) analyzed earlier in De Neve and Van Mieghem (2000). TAMCRA returns the path between a given source and destination subject to the (end-to-end) constraints L_j on each QoS measure ($1 \leq j \leq m$). TAMCRA 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 $w_j(P) = \{w_1(P), w_2(P), ..., w_m(P)\}$ is

[1] Additional results are presented in Kuipers (2000).

the vector sum of the link weights along this path, thus, $w_j(P) = \Sigma_{i \to l \in P} w_j(i \to l)$ for each j. The path length is a vector norm and given by

$$l(P) = max_{1 \le j \le m}(w_j(P)/L_j) \quad (1)$$

This definition (1) obeys the criteria for 'length' or 'distance' in vector algebra (see appendix) and is motivated by the geometry of the constraints surface in m-dimensional space. In the next subsection, other definitions than (1) are considered. An important corollary (see appendix) 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[2] (Henig, 1985). TAMCRA possesses tunable accuracy (coupled to the running time) via one integer parameter k, which reflects the maximum number of shortest paths stored in each nodal queue. There always exists a finite value of k, which is upper bounded by

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

and for which TAMCRA returns the exact path. 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, 1998a).

SAMCRA, Self-Adaptive Multiple Constraints Routing Algorithm, is an exact modification and improvement of TAMCRA that avoids the explicit determination of the parameter k. Instead of requiring in each node of the graph G a queue size k_{max}, the queue size can be adapted independently per node during the course of the routing algorithm. In this mode, TAMCRA self-adaptively finds the exact solution avoiding the need to determine an overall queue size of length $k \le k_{max}$ for each node. In the appendix, we proof that SAMCRA is exact. Besides the concept of dominated paths, an additional increase in the computational efficiency can be gained. Better than checking the path length in an intermediate node with the constraints, each new entry in the queue

[2] A path Q is dominated by a path P if, for all vector components j holds that $w_j(P) \le w_j(Q)$.

is now first compared against the minimum entry in the destination-queue. Indeed, if we already have a path P_1 with length $l(P_1)$, we can discard all the other (sub)-paths P_2 with length $l(P_2) > l(P_1)$, because their length can never be smaller than $l(P_1)$ as the link-vectors consist of non-negative additive metrics.

2.1 SAMCRA implemented with other definitions of length than (1).

Depending on the specifics of a constrained optimization problem, SAMCRA can be used with different length functions, provided they obey the criteria for length. This subsection exemplifies the use of a different definition of length to solve the Delay-Constrained Least-Cost (DCLC) path problem and the Hop-Constrained Maximum Bandwidth (HCMB) problem.

Delay-Constrained Least-Cost (DCLC) path:

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

A suitable length function for this problem is:

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

where $C = N \max_{1 \le i \le E}(c(e_i))$ is a cost-constraint that each cycle-free path can obey, $c(P)$ is the cost of P and $d(P)$ is the delay of P. The length function (5) only optimizes for one metric, the cost or price, which 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. Guo and Matta (1999) have successfully used this approach to solve the DCLC problem. Korkmaz and Krunz (2001) have adopted a similar approach and evaluated the performance of TAMCRA, the predecessor of SAMCRA, and an altered TAMCRA algorithm against Jaffe's algorithms (Jaffe, 1984) and one of the approximation schemes proposed by Hassin (1992). In their simulations, the performance of the TAMCRA(-like) algorithms significantly outperformed the other algorithms.

Hop-Constrained Maximum-Bandwidth (HCMB) problem:

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

A suitable length function for this problem is:

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

where *BW(P)* is the capacity of *P* in minimum units (e.g. Mb/s) and *h(P)* is the number of hops taken by *P*. The length function (6) is similar to (5) and illustrates that min/max parameters can also be incorporated. Besides a different length function, this problem requires 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.,* (RFC 2676, 1999).

We have shown that, provided a suitable definition of length is chosen, SAMCRA applies to numerous constraint/optimization problems.

2.2 The meta-code for SAMCRA

SAMCRA(*G*, *s*, *d*, *L*)

G: graph, *s*: source node, *d*: destination node, *L*: constraints

```
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 grey
8     if(u = d)
9           STOP
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
25                            if(v = d and LENGTH < endvalue)
26                                  endvalue = LENGTH
```

Line 1 to line 4 are initializations, line 1 initializes the counter for each node to zero. This counter keeps track of the number of entries in the queue. 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*, 1991) in line 6 selects the minimum path length in the queue and returns the associated node *u* with entry number *i*. This is the *i*-th path stored in the queue at node *u*. The extracted node is marked gray in line 7. If the extracted node *u* equals the destination, the shortest path satisfying the constraints is returned else the scanning procedure is invoked (lines 11 and 12). 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. The efficiency gain via endvalue in line 16 keeps track of the smallest path length in the destination queue so far. Lines 17 to 24 describe how path(length)s for a node can be added. The endvalue is updated in line 26.

2.3 Worst case complexity of SAMCRA

If N and E are the number of nodes and of links respectively in the graph $G=(N,E)$, the queue in SAMCRA can never contain more than kN path lengths. When using a Fibonacci heap to structure the queues, selecting the minimum path length among kN different path lengths takes at most a calculation time of the order of $\log(kN)$ (Cormen *et al*., 1991). As each node can at most be selected k times from the queue, the EXTRACT_MIN function in line 6 of SAMCRA's meta-code takes $O(kN\log(kN))$ at most. The for-loop starting on line 11 is invoked at most k times from each side of each link in the graph. Calculating the length takes $O(m)$ when there are m metrics in the graph while verifying path dominance takes $O(km)$ at most. Adding or replacing a path length in the queue takes O(1). The total complexity of the for-loop now becomes

$2kE(O(m)+O(km)+O(1)) = O(kEkm)$. Adding the contributions yields a worst-case complexity with $k=k_{max}$ of

$$O\left(kN\log(kN)+k^2mE\right) \tag{3}$$

Clearly, for a single constraint (m=1 and k=1), the complexity (3) reduces to that of the Dijkstra algorithm. When the constraints/metrics L_i are real numbers, the granularity is infinitely small implying that the first argument in (2) is infinite and, hence, $k_{max} = O(N!) = O(exp(NlnN))$. In this case, the QoS routing problem is NP-complete. If L_i for i = 1, …, m can be upper bounded by a constant or polynomial in $\log(L_i)$, (3) is polynomial as proved by Chen and Nahrstedt (1998).

2.4 An example of the operation of SAMCRA

Consider the topology drawn in Figure 2. We are asked to find a path from the source node *A* to the destination node *B* subject to the constraints vector *L = (10,10).*

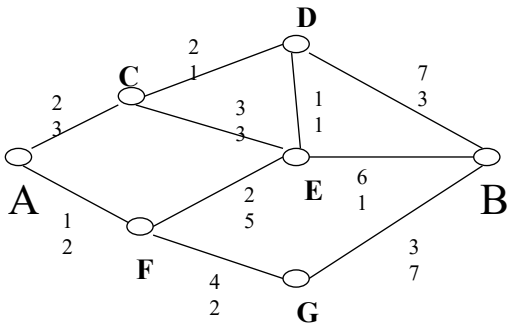

Figure 2. Example topology

SAMCRA returns the shortest path satisfying the *L* vector in 8 steps. Whenever a node is extracted from the queue (line 6 of the meta-code), the corresponding box is colored in gray. The arrows refer to line 11, while line 12 has already been taken into account (i.e. no arrow to an adjacent, previous hop of the path). The algorithm stops when the first entry of the destination node B is extracted from the queue.

In step 4 of Figure 3, the end-to-end length of 1.1 fails the constraint test in line 16 and is not maintained in the SAMCRA queue of node B. Moreover, at step 4, the queue in node E contains two path entries and the new entry: AFE with path vector (3,7) and length 0.7, ACE with path vector (5,6) and length 0.6 and ACDE with path vector (5,5) and length 0.5. Since the vector components of the path ACE are all larger than or equal to those of the path ACDE, the path ACE is dominated by path ACDE and removed from the queue. (The path is not actually removed, but marked to be replaced by the new entry.)

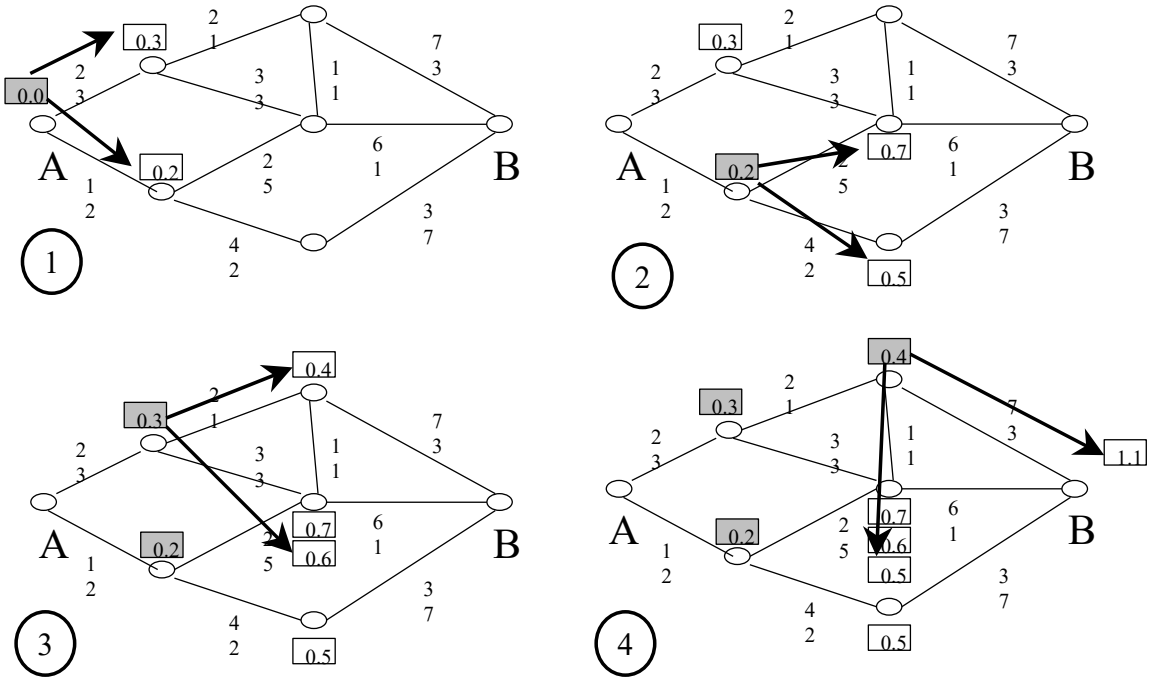

Figure 3. Example of the operation of SAMCRA (step 1 to 4).

Step 5 deserves attention because, since only the previous hop is maintained (line 12), SAMCRA stores 3 new subpaths. But, the path ACDEC with length 0.8 obviously contains a loop. In the dominance check (line 15), this path is again removed since it is clearly dominated by AC. An alternative approach could be to check the entire path so far and only visit neighbors that do not already belong to the subpath. In that case, step 5 would immediately exclude the subpath ACDEC. However, regarding computational complexity, the current approach is more economical if $m<N$, which is normally the case. In step 7, we note that the path AFED with length 0.8 is dominated by the path ACD with length 0.4 and thus not stored in the queue of node D. The only remaining queue entry is at the destination node B and SAMCRA stops in step 8 (line 9). The resulting path satisfying all constraints is AFEB with path vector (9,8) and length 0.9.

Since the granularity is 1 (the vector components are all integers), we observe that, although with (2) k_{max} = 10, k_{min} =*2* suffices for the exact solution because two queue entries are needed at node E, while all the other nodes store equal or less entries. This example also shows that the end-to-end shortest path (A-F-E-B) with length 0.9 is merely the third shortest path from A to the intermediate node E with length 0.7. In addition, this shortest path A-F-E-B is the only path that meets all constraints. If TAMCRA were used with $k = 1$, no path satisfying the constraints would have been found. SAMCRA always guarantees that, if there is a compliant path, this path is certainly found. Finally, the efficiency gain (line 25, 26) only occurs in the final step 8 when we

first find a path satisfying all constraints. Removing (marking black) all queue entries larger than 0.9, unfortunately, does not result in a speed-up in this example.

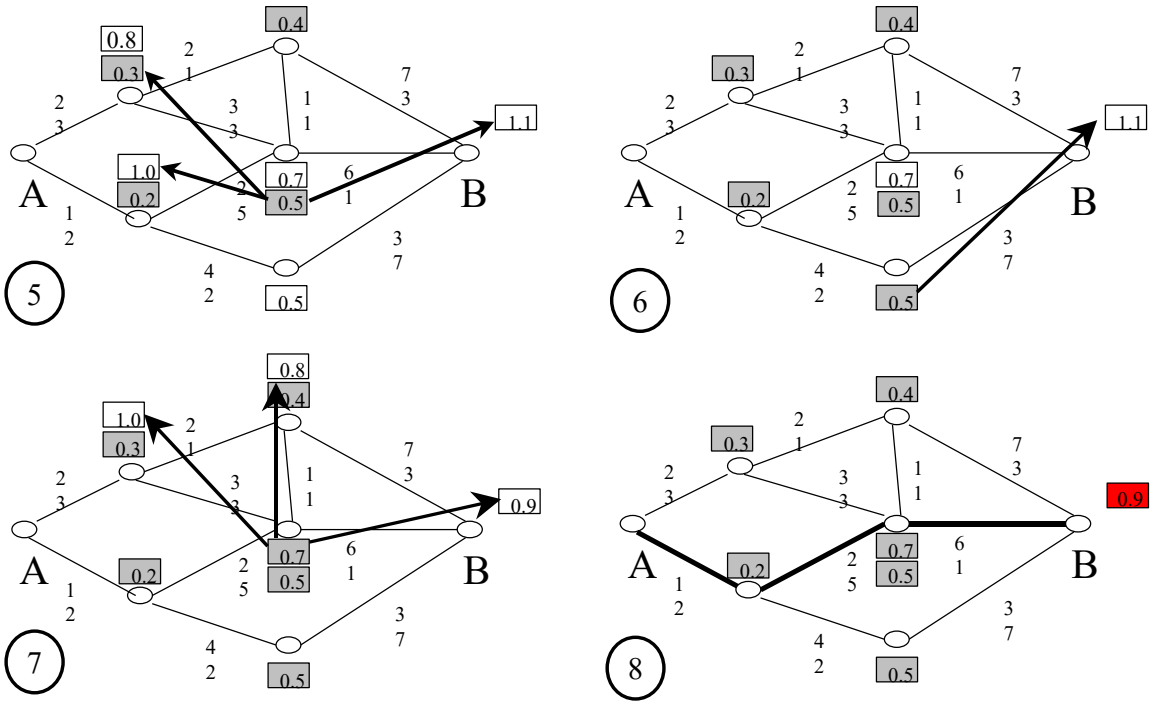

Figure 4. Example of the operation of SAMCRA (step 5 to 8).

3. Hop by hop QoS routing

Since QoS routing assumes multiple end-to-end constraints, a pre-computed routing table should contain in every node at least one path (e.g. the 'shortest' depending on some 'length' criterion) for each source-destination pair and each class of service. This means that every node should store the next hop towards the destination. The next hop is obtained in every intermediate node along the path via a route computation from the *original source* (not the intermediate node itself) to the destination. The number of table entries grows like $O(N^2)$ as opposed to $O(N)$ for ordinary BE where only the destination matters. Zhang et al, (1997) have briefly mentioned two arguments for source-destination routing in unicast Q-OSPF: (a) resource reservations are generally based on a source-destination pair and (b) flows with same destination but different sources can follow different paths. Based on our experience with T(S)AMCRA, there is a more fundamental reason[3] to support source-destination routing embodied by the above mentioned

[3] Since SAMCRA is exact, the corollary is not just an artefact of SAMCRA, but a property of multiple constraints routing in general.

corollary: *when using a non-linear definition of the path length, the subsections of shortest paths in multiple dimensions are not necessarily shortest paths.* Indeed, suppose we construct the end-to-end path in the 'traditional' (as e.g. in OSPF) hop-by-hop way ignoring the source address. This corollary demonstrates that, even with exact multiple parameter routing algorithms (such as SAMCRA), a hop-by-hop route computation does not necessarily result in *the* shortest path, as exemplified below in sec. 3.2. Immediately, the question arises whether a non-linear vector length introduces the possibility of creating loops in the path when computed in a hop-by-hop fashion.

3.1 On routing loops.

Let P_k denote the shortest path from an intermediate hop k towards the destination computed by SAMCRA.

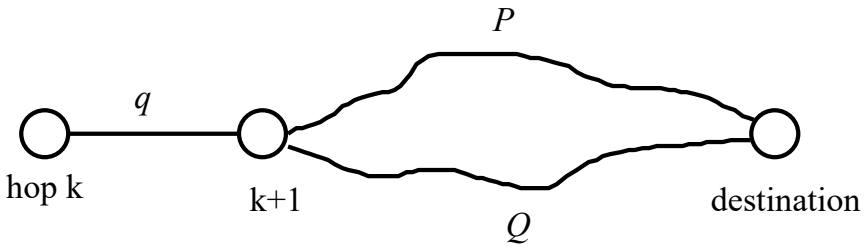

Figure 5 Construction of hop-by-hop paths via SAMCRA

Lemma 1. If P_{k+1} *is a subsection (or sub-path) of* P_k *there holds that* $l(P_k) \geq l(P_{k+1})$ *otherwise* $l(P_k) > l(P_{k+1})$

Proof:

If P_{k+1} is a subsection (or sub-path) of P_k, we have that $P_k = q + P_{k+1}$ where q is a single link vector (the first link vector on the shortest path from hop k to the destination). The first condition then is immediate from (A.2) in the Appendix.

The second condition follows from the corollary that subsections of shortest paths are not necessarily shortest paths. Hence, we may have that, referring to Figure 5, the path $P_k = q + Q$ is the shortest path from k to the destination while $P_{k+1} = P$ is the shortest path from hop $k+1$ to that same destination. Thus, $l(P + q) > l(Q + q)$ and $l(Q) > l(P)$. Using (A.2) gives $l(Q + q) \geq l(Q)$. Combining all inequalities leads to $l(P + q) > l(Q + q) \geq l(Q) > l(P)$ or $l(P_k) > l(P_{k+1})$. QED.

Lemma 1 implies that the sequence of the lengths of the shortest paths from the different intermediate hops to the (same) destination is non-increasing: $l(P_1) \geq l(P_2) \geq \ldots \geq l(P_k) \geq l(P_{k+1}) \geq \ldots \geq 0$. The end-to-end path from source node S to destination node D computed by hop-

by-hop SAMCRA consists of the node list *{S, n_2, ..., n_j, ...,D}* where n_j is the second node in the node list of path P_{j-1}.

Property 1. *The end-to-end path computed by hop-by-hop SAMCRA is guaranteed to be loop-free*

Proof:

Property 1 is a consequence of lemma 1. Indeed, a loop means that at some hop k, the path has returned to a node n previously visited at hop j with $j < k$. Further, the presence of a loop implies that not all P_k are subsections of P_1 and that, in the sequence of lengths of shortest paths between hop j and hop k, there must be at least one strict inequality sign (lemma 1) yielding $l(P_j) > l(P_k)$ with $j < k$. Let P denote the shortest path from node n to the destination. At hop j, we first arrive at node n and we have $P_j = P$. Since the loop returns during hop k at node n again, we obtain $P_k = P$ implying that $P_j = P_k$ and thus $l(P_j) = l(P_k)$ for $j < k$. This condition contradicts the previous one based on lemma 1. QED.

Corollary. *Removing dominated paths Q for which holds that $w_j(P) \leq w_j(Q)$ and thus $l(P) \trianglelefteq l(Q)$, also eliminates loops.*

In conclusion, we have demonstrated that 'hop-by-hop destination based only' multiple constraint routing with SAMCRA is loop-free. In case of an approximate QoS routing algorithm that cannot guarantee to find always the shortest path towards the destination in every hop, the arguments used in the proof of property 1 and lemma 1 do not exclude the occurrence of loops. Hence, so far, we have shown that exactness is a sufficient condition. It remains to show that exactness is also necessary. Therefore, it suffices to illustrate with an example that loops can indeed occur if the *x-th* shortest path is chosen with $x > 1$.

Consider the simple graph of Figure 6 where a path is to be found from source A to destination B subject to a constraints vector L = (13,13). If the source node A uses an exact computation for the QoS path towards B, it finds two paths: A-D-B with length max(4/13,12/13) = 12/13 and A-C-D-B with length max(9/13,11/13) = 11/13 which is clearly the shortest path from A to B. Hence, source node A forwards the packet for B to C. Suppose now that node C uses an approximate algorithm, such as TAMCRA with $k = 1$. Node C receives the packets without remembering where they came from (i.e. 'hop-by-hop destination based only') and it computes the path towards B as follows (very similar to Dijkstra's greedy algorithm). First the adjacent nodes D and A are investigated with length 6/13 and 1/13 respectively. Second, the computation proceeds with

A (the smaller of the two length so far) and we find that the path C-A-D has length 5/13. Now, since C computes paths with an approximate QoS algorithm where *k=1*, it only stores the shortest path to node D, which is C-A-D. Hence, C computes the path C-A-D-B with length 1. However, although the path C-A-D-B just satisfies the constraints, it clearly creates a loop because the packet is sent back to the source node A.

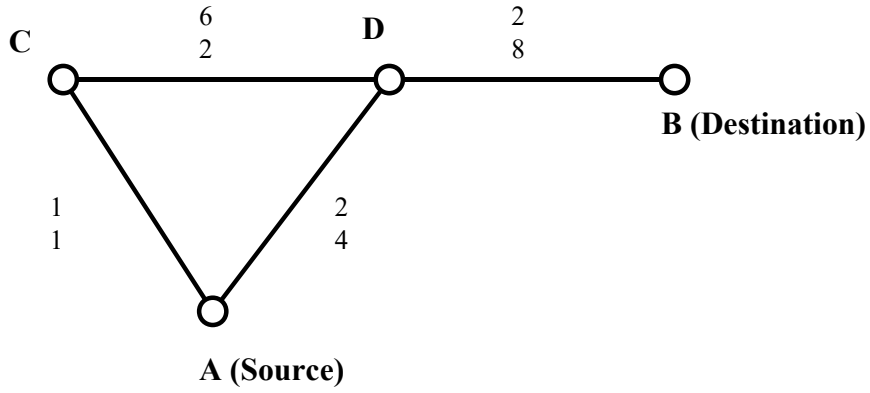

Figure 6 Example demonstrating the existence of a loop with approximate QoS routing.

This example shows that approximate algorithms can produce loops by excluding intermediate paths. In fact, C has stored the second shortest path to B instead of the shortest. If C had used TAMCRA with *k=2*, the correct path (without loops) would have been found. Indeed, both paths C-D and C-A-D would have been checked, which would have resulted in C-D-B with length 10/13 instead of C-A-D-B with length 1. In that case, C would have forwarded the packet for B to D, without any bouncing (or loops).

3.2 HbHDBO QoS routing cannot guarantee to find the exact result.

We now investigate the quality of QoS routing relying on a hop-by-hop approach only based on the destination. In particular, only the shortest path (according to SAMCRA's non-linear length definition) is computed in each hop towards the destination (ignoring the previous history where it came from). The resulting end-to-end path computed in HbHDBO mode via SAMCRA is compared, as shown in Figure 1, with the shortest end-to-end path satisfying all QoS constraints.

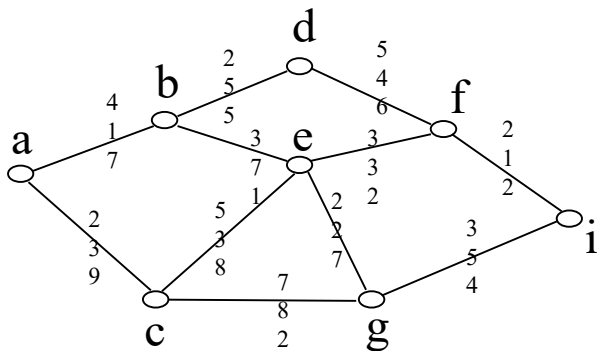

Figure 7 Example topology

The following example topology (Figure 7) demonstrates that (exact) SAMCRA operating in a distributed HbHDBO mode cannot guarantee to find the exact result.

Each link is characterized by three additive QoS metrics (or vector components). Suppose that node *i* wants to send packets to node *a* according to a certain CoS. The CoS is characterized by a constraint for each of the graph metrics. In this example, the constraints are chosen to be 14, 11 and 22 respectively. Using SAMCRA, node *i* calculates the shortest path to node *a* as shown in Figure 8.

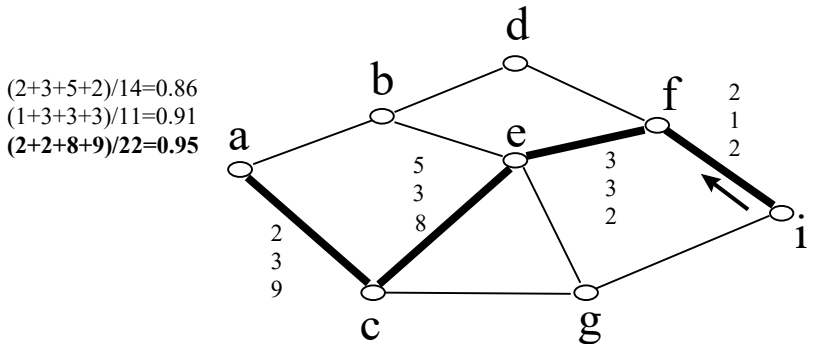

Figure 8 Shortest path from node *i* to node *a*.

The shortest path from node *i* to node *a* runs over nodes *f*, *e* and *c* and has a path length of 0.95. Thus, all constraints are satisfied. Node *i* will store in its routing table that the next hop for destination *a* and the class of service specified by constraints (14, 11, 22) is node *f*. Node *f* will construct its routing table in a similar manner: it uses SAMCRA to calculate the shortest path to destination *a* subject to the constraint vector (14, 11, 22). So far, no deficiencies have occurred for the packets originated at node *i* because the shortest path from node *i* to node *a* coincides with the shortest path from node *f* to node *a*.

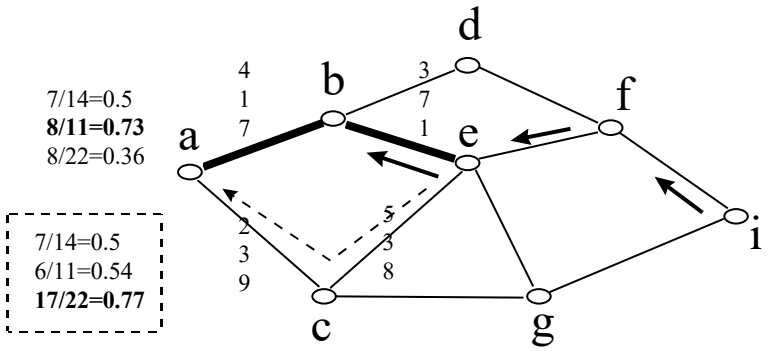

Figure 9. Constrained shortest path form node *e* to node *a*

However, this is no longer the case for node *e*! The shortest path from node *e* to node *a* for the class of service characterized by the constraint vector (14, 11, 22) is shown in Figure 9. The shortest path from node *e* to node *a* runs over node *b* and not over node *c*. The path over node *c* is shown with dashed lines and has a path length of 0.77 which exceeds the path length of the path *e-b-a* which is only 0.73. Thus node *e* will store in its routing table that all packets for destination *a* and the above specified CoS need to be forwarded to node *b*. Packets from node *i* will thus no longer follow the shortest path between node *i* and node *a*.

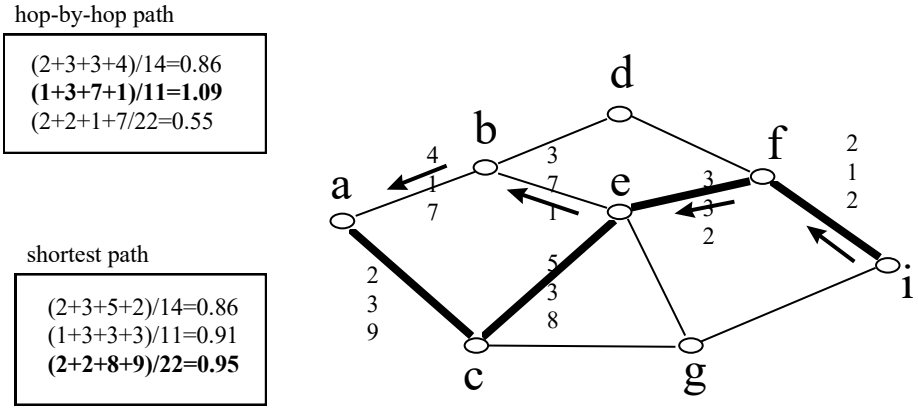

Figure 10. Comparison between the shortest path (in bold) and the hop-by-hop path (arrows).

Figure 10 shows both the end-to-end hop-by-hop path and the shortest path from node *i* to node *a* for the CoS specified by the constraint vector (14, 11, 22). The shortest path satisfies all constraints but the hop-by-hop path has an unacceptable path length of 1.09 > 1. In particular, the box in Figure 10 indicates that the hop-by-hop path satisfies the constraints for the first and the third metric but violates the constraints for the second metric. Thus, although there is a path from node *i* to node *a* satisfying all constraints, in a distributed hop-by-hop mode, this example illustrates that the packets will follow a path that violates one of these constraints.

In this example, it can be verified that the hop-by-hop path turns out to be the third shortest path from node *i* to node *a* for the CoS specified by constraints (14, 11, 22). The second shortest path is *P(i, f, d, b, a)*. The path length of the second shortest path satisfying all constraints is exactly 1.00, which means that at least one of the constraints is precisely met.

4. Performance Analysis in Simple Random Graphs.

The simulations were performed on (simple) random graphs of the class $G_p(N)$ (Bollobas, 1985) consisting of N nodes and independently chosen links with probability p. Each link is further specified by a vector with m components, which are all uniformly distributed random variables on [0,1]. We further denote the class $G_p(N)$ with uniformly distributed link weights by RGU. In each graph of RGU, the shortest path between source (node 1) and destination (node N) subject to a constraints vector with components $L_j = N$ (for all $1 \le j \le m$) has been computed with SAMCRA in the two modes as shown in Figure 1.

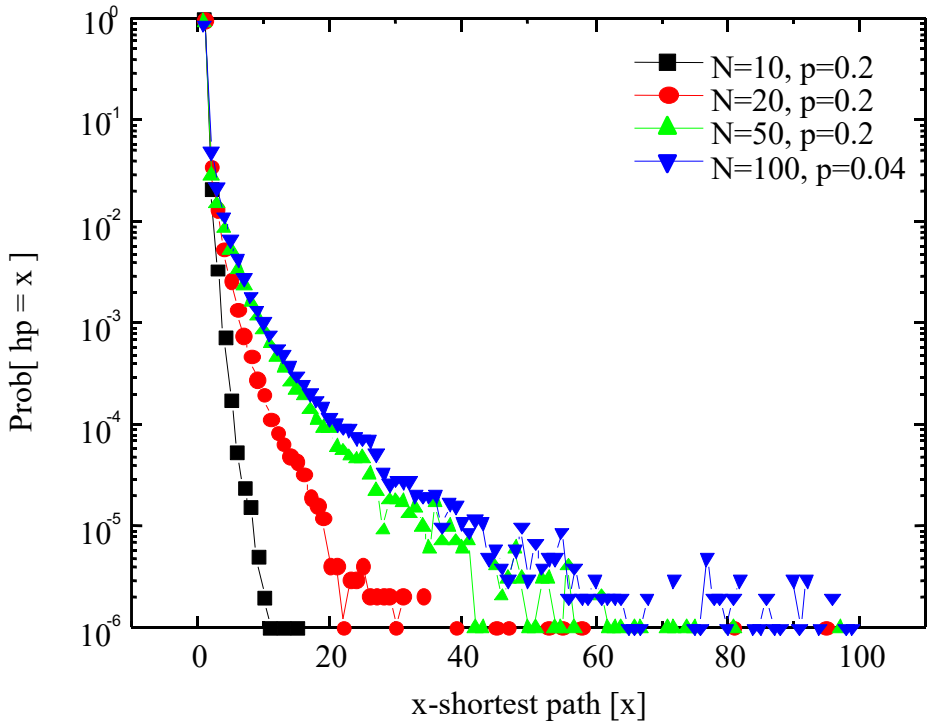

Figure 11 The pdf of the ranking in shortestness of the hop-by-hop (hp) shortest path for $m=2$

The value $L_j = N$ for each constraint j implies that all paths from source to destination lie within the constraints. Via the exact source-routing mode, a list of all possible shortest paths was computed as reference to compare the HbHDBO path. For each quantity of interest such as the *x*-

th shortest path, difference in the length etc., one million RGU graphs were simulated, a number satisfactorily large to obtain accurate probability density functions up to roughly 10^{-4} for that quantity. Similar to one-dimensional shortest path problems (see Van Mieghem *et al.,* 2000), the dependence of the shortest path properties to the link density p (above the disconnectivity threshold) in higher dimensions also was found negligibly small. This presumably general law for the class of RGU allows us to eliminate the extensive simulations as function of p. Besides the ease in computing a graph of the RGU class, its use was mainly motivated by an analogy to linear system theory. By applying an impulse to a linear system, all internal modes are excited. Similarly, by simulating a large number of different graphs in RGU, the underlying randomness was expected to trigger the different 'modes' of the hop-by-hop QoS routing problem. We emphasize, however, that all simulation results only apply to the class RGU.

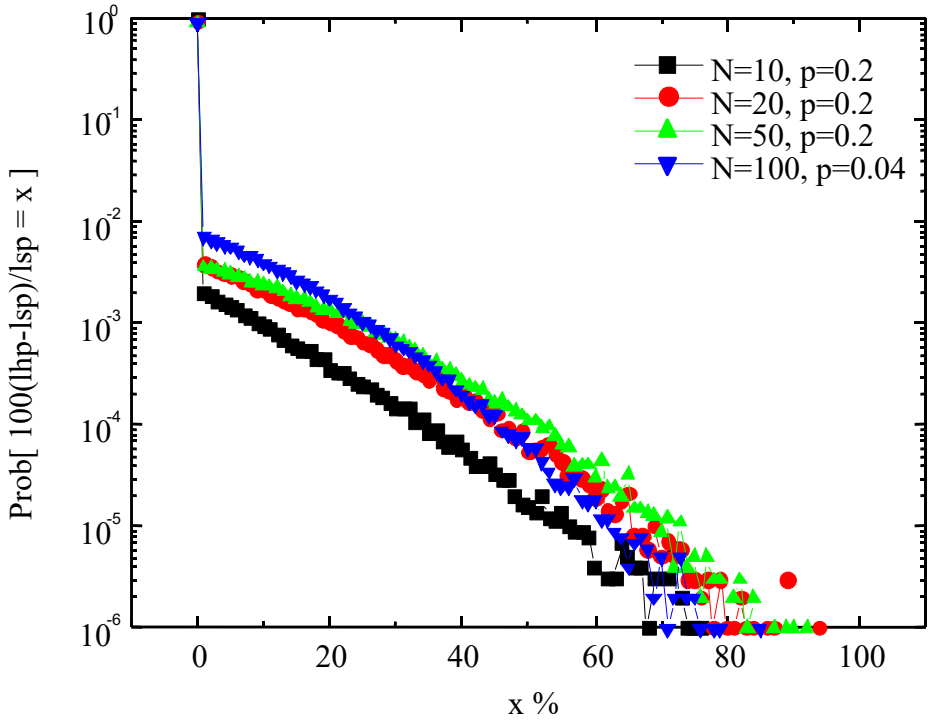

Figure 12. The probability that the length of the HbHDBO path is x% longer than the shortest path for *m=2*

4.1 The x-th shortest path found with HbHDBO QoS routing.

For various number of nodes N and link density p, Figure 11 plots the probability that the HbHDBO path is the x-th shortest path. The exact solution ($x=1$) is found in 89.4% of the cases in RGU with $N=100$. When the network size N increases, the probability that the exact solution is found decreases as expected because the number of paths between source and destination

increases with N and, hence, also the average hop count and the average number of decisions which enhances the chance on wrong decisions. When constraints ($L_i < N$-1) are imposed, the number of paths between source and destination will increase, of course, less rapidly.

4.2 The difference in end-to-end path length.

Figure 12 shows that, although the exact path (x=1) may not be found, the relative difference in length between the HbHDBO and shortest path in the class RGU is surprisingly small (for x>0 below a probability of 0.01) and that it decreases faster than an exponential. In any topology, property 2 upper bounds the relative error in case one wrong decision is made.

Property 2: *If the HbHDBO path only makes one wrong decision (compared to the true shortest path), the resulting length never exceeds 100% of the length of the (exact) shortest path. Moreover, this bound is sharp.*

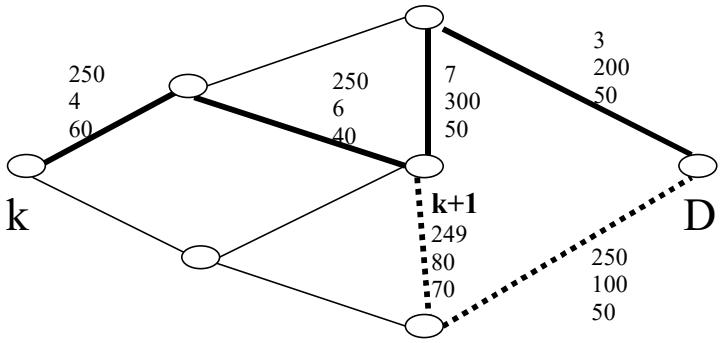

Figure 13 The shortest path is drawn in bold, whereas the HbHDBO path is in dotted line. The constraint vector is (1000,1000,1000). Referring to the notation of previous Figure, we have that *l(Q)=0.499, l(P)=0.5, l(q+P)=0.510* and *l(q+Q)=0.999* resulting in a relative difference of 96%.

Proof:

Consider the graph in Figure 5. Assume that the shortest path from source k to destination d consists of the paths q and P with length $l(q + P)$ while the shortest path from $k+1$ to d is Q. Thus, $l(Q) < l(P)$. Ignoring the past history, the hop-by-hop path makes a wrong decision at node $k+1$ where it follows path Q resulting in a final path with length $l(q + Q) > l(q + P)$.

Using the properties of the length of a vector (see appendix), we can arrange the lengths in increasing order as follows:

$l(Q) < l(P) \leq l(q + P) < l(q + Q) \leq l(q) + l(Q) < l(q) + l(P) \leq l(q) + l(q + P)$

The relative difference in length therefore has an upper bound:

$$\frac{l(q+Q)-l(q+P)}{l(q+P)} < \frac{l(q)+l(q+P)-l(q+P)}{l(q+P)} = \frac{l(q)}{l(q+P)} \le 1$$

Figure 13 demonstrates that this bound is sharp. QED.

The proof shows that a wrong hop decision at the end of the path (near the destination) can have more severe consequences than a wrong decision near the source (the first hop is always correct!), because *l(q)* tends to *l(q + P)* when we hop towards the destination. An extension of property 2 to multiple wrong decisions is straightforward resulting in an overall upper bound equal to the sum of the independent, single, wrong hop upper bounds.

4.3 Simulated complexity of SAMCRA in the class RGU

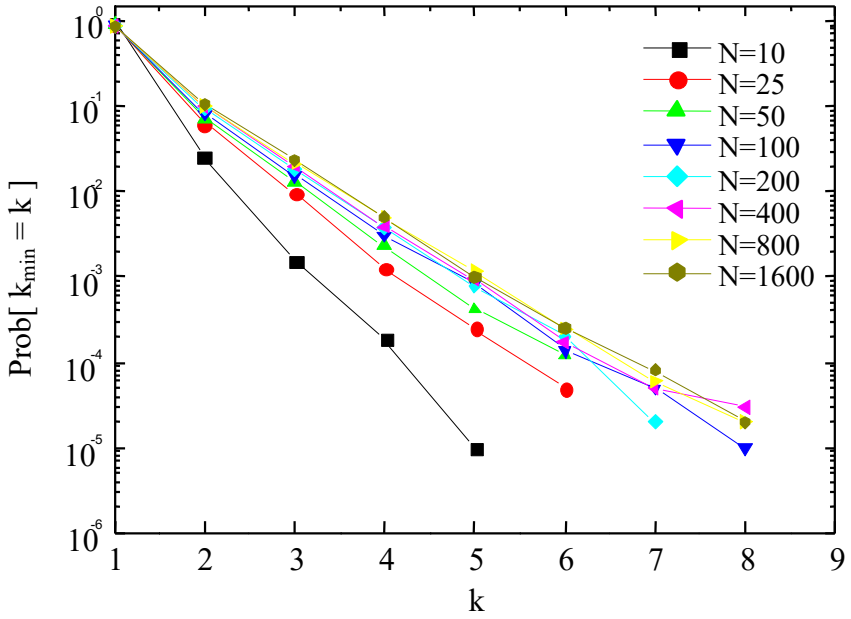

Figure 14. The probability density function of k_{min} . Note that E[k_{min}] ≈ 1 for N up to 1600 and that max(k_{min}.) does not exceed 10.

Figure 14 plots the minimum queue size needed to find the exact path. This minimum queue size k_{min} is, in fact, the maximum queue size that appears in a node during the evaluation of SAMCRA. If a lower queue size $k < k_{min}$ were used, the exact shortest path would not be found. The constraints in the simulation-program were denoted as doubles (real numbers), which makes the upper bound on the queue-size k_{max} (2) extremely large. With an expected E[k_{min}] ≈ 1, these simulations suggest that the average complexity for the class of RGU is *O(NlogN+mE)*.

4.4 The influence on the number of constraints m

Figure 15 and Figure 16 show the influence of the dimension *m* on the quality of the HbHDBO path. As well known (see also Corollary 1 in the Appendix) for a single metric or *m=1*, the HbHDBO path is also the exact shortest path. In higher dimensions (*m*>1) and in the class RGU, we observe that the case with *m=2* is worst and that there is a noticeable decrease towards the exact solution as *m* increases.

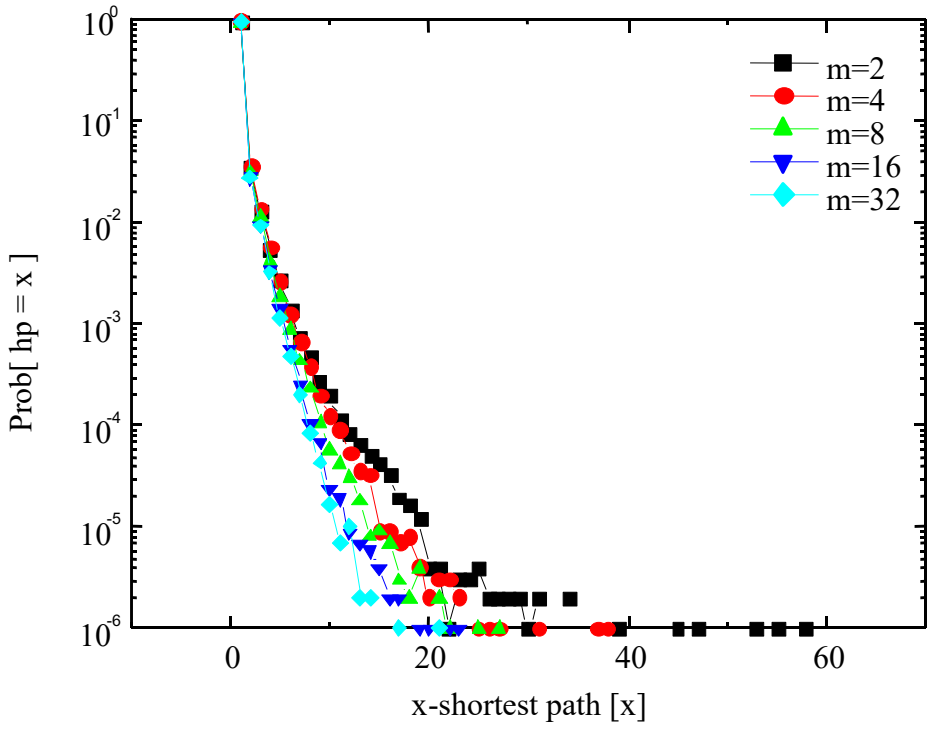

Figure 15. Probability that the hop-by-hop path equals the x-shortest path in various dimensions m for N=20 and p=0.2.

Property 3: *If the m components of the link weight vector are independent random variables and the constraints L_j are such that $0 \le w_j/L_j \le 1$, any path with K hops has precisely a length (as defined in (1)) equal to K in the limit $m \to \infty$.*

Proof:

Consider a path *P* from source to destination with *K* hops. The vector sum of the weights along this path equals $w_j(P) = \Sigma_{i \to l \in P} w_j(i \to l)$ and $0 \le w_j(i \to l)/L_j \le 1$. Then $Prob[w_j(P)/L_j > K]=0$. Moreover, with definition (1), $Prob[l(P)>K] = 0$. Since we assume that the link weights are independent, we have $Prob[l(P) \le K-\varepsilon] = \Pi_{1 \le j \le m} Prob[w_j$

(P)/$L_j \le K$-ε]. But, for any real *ε>0* and each *j*, *Prob[w_j(P)/$L_j \le K$-ε])<1*, *Prob[l(P) $\le K$-ε] = 0* for $m \to \infty$. Hence, in that limit, each path with *K* hops has length precisely equal to *K*. QED.

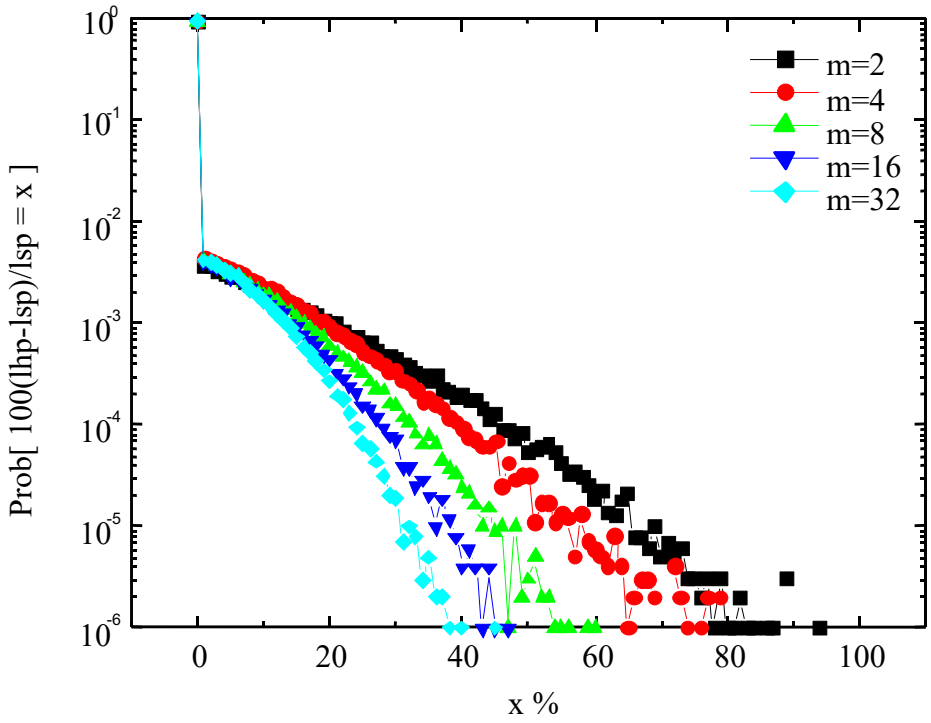

Figure 16. The probability that the relative length difference is x% for N=20 and p=0.2.

For $m \to \infty$, the *m*-dimensional problem reduces to a single metric problem where the path that minimizes the hopcount is also the shortest path. As the latter problem is exactly solvable (via Dijkstra), this explains the observed decrease towards the exact solution as *m* increases. Notice that the limit law in property 3 is quite general: It only assumes independent link weight components w_j and constraints L_j such that $0 \le w_j/L_j \le 1$ and it holds for any graph.

5. Active Networking

The model assumption of pre-computed routing tables as in current Internet, is actually limiting the flexibility and quality of hop-by-hop QoS routing. Active networking, a new dynamic concept in networking, can provide the necessary flexibility to guarantee QoS in hop-by-hop routing. In active networks, the active nodes can perform computations on the data part of the datagram and the users, on the other hand, can "program" the network by supplying the active node with their

instruction code [Tennenhouse et al., 1996]. Two 'active' scenarios for hop-by-hop QoS routing with SAMCRA are discussed:

1. The first scenario will be referred to as "active constraints". Similar as in Q-OSPF (RFC2676), the routing table at a node contains the next hop (computed with SAMCRA) for each class of service (CoS) specified by a constraint vector L_{CoS} for that node. Each packet contains the 'active' constraints vector L which is decreased at each node by the link weight vector of the previous hop. The packet is classified (and accordingly forwarded) in that CoS for which $L_j > L_{CoS,j}$ for $1 \leq j \leq m$ and for which the next upper CoS, say CoS', has at least one component j violating the condition $L_j > L_{CoS',j}$. This type of active routing excludes forwarding over paths that violate the constraints, without adding much complexity: O(Nm) per packet, because there are not more than N-1 hops and at each node there are m subtractions.
2. In the second scenario (called "active SAMCRA") the packet has an extra field containing the link weight vector of the intermediate path traversed so far and the original (fixed, not decreased) constraint vector L. At each hop, the link weight vector and the path towards the destination is computed. Active SAMCRA is exact as can be verified from SAMCRA's meta-code at the price of a higher complexity: O($kNlog(kN) + k^2mE + m$) per packet per node.

Both scenarios guarantee that a path within the constraints is returned *if such a path exists*. The difference between "active constraints" and "active SAMCRA" is that the first might return non-shortest (feasible) paths, whereas the second *always* returns the shortest path. Since our goal is to find a path within the QoS constraints, the first approach prevails over the second, due to its smaller complexity. However, if we consider a rapidly changing topology (due to coupling of the link weights with the traffic in that link for instance), active SAMCRA is to be preferred because computing one path costs less than computing a whole routing table at each frequently occurring topology update.

6. Summary.

We have investigated the problem of connectionless QoS routing. To limit the number of routing entries, we have investigated the possibility of forwarding packets based on the destination only. We have first demonstrated that an exact algorithm as SAMCRA is loop free when used in hop-by-hop mode. Moreover, with a non-linear path length definition, an exact algorithm is not only sufficient but also necessary to avoid loops. We have further shown that

"hop-by-hop destination based only" QoS routing, even with an exact routing algorithm (and contrary to 'single metric routing' as in OSPF), does not guarantee that the exact end-to-end solution is found. Extensive simulations in the class of RGU have shown that deviations from the exact results are small and occur infrequently. Moreover, the maximum relative difference in length can be bounded.

The intractability of exact QoS routing with multiple additive link measure (due to NP-completeness) is only one of the complicating factors in the QoS routing problem. Indeed, topologically SAMCRA does not scale much worse than Dijkstra. Moreover, simulations in RGU (up to N=1600 nodes) indicate that the minimum queue size needed to retrieve the exact result is much smaller than the worst case prediction since $k_{min} << k_{max}$. In spite random graphs do not resemble actual communication networks, we believe that future topologies due to the increased importance of wireless and ad hoc (or imbedded) networks, may possess a larger degree of randomness both in topological structure as in the link weights. However, mainly the granularity of the constraints as exhibited by expression (2) for k_{max} determines the complexity. Hence, a careful design of QoS routing (thus both algorithm and protocol) may benefit from a coarse granularity in the constraints (i.e. decreasing k_{max}). Also the dynamic behavior of QoS routing (not considered here, but see Van Mieghem and De Neve, 1998) simplifies as the number of topology updates triggered by significant changes in the link metrics diminishes accordingly. Thus, we believe that SAMCRA together with a minimum acceptable k_{max}, is a promising strategy to provide a notion (hard guarantees are shown to be impossible) of QoS in the hop-by-hop Internet of today.

Finally, if the current Internet mode of HbHDBO routing (ignoring the past path history) is relaxed, at the expense of more computations, hop-by-hop routing with active packets can be exact provided the length vector of the path traversed so far and the constraint vector is stored in each packet. This observation is directly verified from the meta-code of SAMCRA.

References

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

Apostolopoulos, G., R. Guerin, S. Kamat and S. K. Tripathi, 1999, " Improving QoS Routing Performance Under Inaccurate Link State Information", ITC16, pp. 1351-1362

Bollobas, B., 1985, Random Graphs, Academic Press, London.

Chen, S. and K. Nahrstedt, 1998, "*On Finding Multi-constrained Paths*", Proc. ICC '98, Atlanta, Georgia.

Chen, S. and K. Nahrstedt, 1999, "Distributed Quality-of-Service Routing in Ad Hoc Networks", IEEE J. Selected Areas in Communications, vol. 17. No. 8, pp. 1488-1505.

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, 1991, *Algorithms*, MIT Press, Boston.

De Neve, H. and P. Van Mieghem, 1998, "A Multiple Quality of Service Routing Algorithm for PNNI", IEEE ATM Workshop, (Fairfax, May 26-29), pp. 324-328.

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

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. A. and A. Orda., 1999, "QoS Routing in Networks with Inaccurate Information: Theory and Algorithms", IEEE/ACM Transactions on Networking, Vol. 7, No. 3, June, pp. 350-364.

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.

Hassin, R., "Approximation schemes for the restricted shortest path problem", Mathematics of Operations research, 17(1): 36-42, February 1992.

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

Jaffe, J.M., "Algorithms for finding paths with multiple constraints", Networks, vol. 14, pp. 95-116, 1984.

Korkmaz, T. and M. Krunz, "Multi-Constrained Optimal Path Selection", INFOCOM 2001.

Kuipers, F. A., 2000, *Hop-by-hop routing with QoS constraints*, Master Thesis in Electrical Engineering at Delft University of Technology.

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

Sales, B. and P. Van Mieghem, 1998, "Dual-mode Routing: A General Framework for IP over ATM integrated Routing", Proceedings of the Third IEEE Symposium on Computers and Communications (ISCC'98), June 30 -July 2, Athens (Greece), pp. 326-330.

Tennenhouse, D.L., S.J. Garland, L. Shrira and M.F. Kaashoek, 1996, "*From Internet to ActiveNet*", RFC January, http://www.tns.lcs.mit.edu/publications/rfc96/.

Van Mieghem, P. and H. De Neve, 1998, "Aspects of Quality of Service Routing", SPIE'98 (Boston, November 1-5), [3529A-05].

Van Mieghem, P., 1998a, "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, 2000, "A Scaling Law for the Hopcount in Internet", Delft University of Technology, report 2000125 (http://wwwtvs.et.tudelft.nl/people/piet/telconference.html).

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

Zhang, Z., C. Sanchez, B. Salkewicsz and E. Crawley, 1997, "Quality of Service Extensions to OSPF or Quality of Service Path First Routing (QOSPF), draft-zhang-qos-ospf-01.

Appendix

The definition $l(.)$ of the length of a vector p, such as (4), satisfies the criteria (Golub and Van Loan, 1983; Royden, 1988),

(1) $l(p)>0$ for all non-zero vectors and $l(p)= 0$ only if $p = 0$.

(2) for all vectors p and u holds the triangle inequality

$$l(p + u) \leq l(p)+l(u) \tag{A.1}$$

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

$$l(p + u) \geq l(p) \tag{A.2}$$

because the length of a non-negative vector cannot decrease if a non-negative vector is added.

For example, if $p = [1, 3, 5, 1, 9]$, $u = [4, 5, 2, 1, 0]$ and $L = [10, 1, 10, 10, 1]$, then, using (1), $p + u = [5, 8, 7, 2, 9]$, $l(p) = 9$ and relation (A.2) gives $l(p + u) = l(p) = 9$.

Corollary. *Subpaths of shortest paths in multiple dimensions (m>1) are not necessarily shortest paths*

This is an important corollary that follows from non-linear path length definitions as (1) or

$$l(p) = \left(\sum_{j=1}^{m} w_j^q \right)^{1/q} \tag{A.3}.$$

The proof relies on the inequality (A.1). Consider two paths P_1 and P_2 for which $l(P_1) < l(P_2)$ and assume that, by adding a same link a to both paths, the paths P_3 and P_4. can be constructed.

1. Let us first focus on the case where the equality sign in (A.1) holds, typically if $q = 1$ in (A.3). By construction, we have $l(P_3) = l(P_1 + a)$ and by (A.1) with the equality sign, $l(P_1 + a) = l(P_1) + l(a)$ and analogously, $l(P_4) = l(P_2 + a) = l(P_2) + l(a)$. Since $l(P_1) < l(P_2)$, there holds that $l(P_3) < l(P_4)$ or, the subpaths of shortest path with linear definition of path length are again shortest paths, leading to the well-known and intuitive result.
2. When the inequality sign holds in (A.1), typically if q > 1 as readily verified from (A.3), we arrive in a similar fashion to the set of inequalities $l(P_4) = l(P_2 + a) < l(P_2) + l(a)$; $l(P_3) = l(P_1 + a) < l(P_1) + l(a)$ and $l(P_3) < l(P_4)$. However, from this set, it cannot be concluded whether $l(P_3) \leq l(P_4)$ or $l(P_3) > l(P_4)$. It suffices to show that the latter situation may exist in order to prove the corollary. This can be illustrated on the example graph of Figure 7. For the constraints (14,11,22), the shortest path from node a to node i runs over node c, e and f as shown in Figure 8. According to the definition (1), the path length of $P(i,f,e,c,a)$ equals 0.95 which means it satisfies all constraints. However, as illustrated in Figure 9, the shortest path from node a to node e does not traverse node c but node b. QED.

Theorem. *SAMCRA is exact*

Proof. SAMCRA is essentially a brute-force algorithm that efficiently reduces its search-space. SAMCRA uses a k-shortest path approach. As shown by Chong *et al.,(*1995), this k-shortest path approach will return all (loop-free) paths between source and destination, provided k is chosen large enough. It can be verified from SAMCRA's meta-code that in

the absence of the search-space reduction, SAMCRA evaluates every possible path from source to destination, and therefore is exact by definition.[4] To prove that SAMCRA in its search-space reduction is exact, we need to prove that the reductions do not exclude the shortest path.

1) Lines 8 and 9 in SAMCRA's meta-code cause the algorithm to stop when the extracted path belongs to the destination node. Since line 6 only extracts the path with the shortest length, all other path lengths left in the queue are equal or longer than the length of the extracted path and hence will never become the shortest path. The shortest path from source to destination has been attained and it is therefore useless to proceed with the algorithm. We have therefore justified lines 8 and 9.
2) SAMCRA discards all paths that exceed the constraints (line 16: `if(LENGTH ≤ 1.0)`). Since these paths cannot accommodate the requested QoS, there is no need to examine them.
3) SAMCRA has two ways to remove loop-containing paths from the search-space:
   a) Line 12 ensures that the path does not return to the node it just came from. This way of preventing loops only eliminates loops between two adjacent nodes.
   b) Loops containing more nodes are targeted by only allowing non-dominated paths (lines 15 and 16) as follows from the corollary in sec. 4.1.
4) Finally, each new entry is compared to the minimum entry in the destination-queue. If we already have a path P_a with length $l(P_a)$, we can discard all other (sub)-paths P_b with length $l(P_a) \leq l(P_b)$, because their length will never become smaller than $l(P_a)$. QED.

[4] The proof that SAMCRA without search-space reduction returns all paths is given in Kuipers (2000).