Introduction to
Shortest–path routing & Quality–of–Service routing

Lars Staalhagen
Networks Incompetence Area
Research Center COM
ls@com.dtu.dk

Version 0.1 — July 21, 2005

1 Introduction

The purpose of this lecture note is to introduce two concepts in routing: Shortest–path routing and Quality–of–Service routing. It is intended to be used in course 34355 Routing in Data Networks.

The lecture note is structured as follows: In section XXXX the concept of shortest–path routing is described, along with two common algorithms for finding the shortest–paths in a network. In section XXXX the Quality–of–Service routing (QoS–routing) is described, starting with a general definition of QoS and QoS routing. In the following two sections, some problems of QoS–routing in relation to multicasting and ad–hoc networks are described.

2 Shortest path routing

The aim of shortest–path routing[1] is to find an optimal path in a network from a source node to a destination node.

The algorithms that are used to determine an optimal path assume a graph–representation of the network, i.e., $G = (N, V)$, where N are the nodes in the graph and V are the edges[2]. Corresponding to a real network, the nodes correspond to the routers, while the edges correspond to the communication links. Since communication links can be either unidirectional (communication is only possible in one direction) or bidirectional (communication is possible in both directions), this must also be represented in the graph. In the first case, the edges include an arrowhead to indicate the direction of the communication; this edge is then said to be a *directed* edge. A graph is said to be *directed* if it includes one or more directed edges; otherwise the graph is *undirected*.

Figure 1 illustrates an undirected and a directed graph.

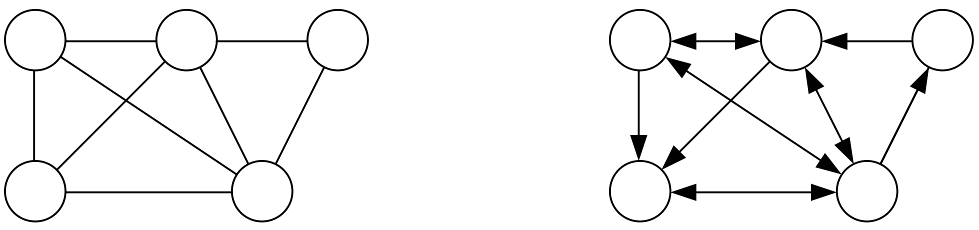

Figure 1: Undirected and directed graph.

Every edge in the graph is associated with a metric (a number) that represents the "length" of that edge, so that the path from a source node to a destination node has a total length of the sum of the length of the edges that are part of the path. In this sense, the link–metrics are said to be *additive*.

[1]In some textbooks the term *least–cost* routing is used instead.
[2]Other names are *arcs* or *lines*

The metrics represent some characteristic of the link in the real network, for instance

- Delay – the metric is (directly) proportional with the delay on the link, so that the shortest–path from the source to the destination is the path that minimizes the end–to–end delay. The link–delay may include any buffering delay at the sending node.
- Economical cost – the metric is (directly) proportional with the expenditure of transmitting information on the link. The shortest–path is then the path that minimizes the expenditure for the network operator.
- Error rate – the metric is a function of the link's error rate[3], so that a link with a high error–rate is depicted in the graph as an edge with a high cost. Finding the path in the graph that minimizes the sum of the edge metrics would then tend to favour links with lower error rates, since they would have a smaller metric.

In general, the link metrics may also be a function of (some of) the parameters of a link, i.e.

$$\text{Metric} = f(\text{delay}, \text{economical cost}, \ldots)$$

The function that is used to generate the metric is chosen by the network operator to reflect his preferences, for instance that the paths determined tend to favor low–delay links and tend to avoid high–cost links.

A number of algorithms exists that can be used to determine the shortest paths, but in connection with communication networks, the most common are the *Bellman–Ford* algorithm and the *Dijkstra* algorithm. The former is used in connection with the RIP protocol, while the latter is used with the OSPF protocol. Note that the results of these algorithms are not just the shortest path from a given source node to a given destination node, but instead a set of shortets–paths, e.g. the shortest–paths from a given source–node to *all* other destination nodes.

In mathematical terms, the link–metrics are here denoted by C_{ij} for the metric on the link *from* node i *to* node j. In general, $C_{ij} \neq C_{ji}$, i.e. for a bidirectional link, the metrics are different in the two directions. This might be the case if the metrics are a function of the link delays. By definition, $C_{ii} = 0$, since (informally) it "costs" nothing to get a packet to node i, when it is already at node i. Furthermore, two different nodes, i and j, might not have a direct edge between them, so it is impossible to send a packet directly from node i to node j. This situation is represented in the metrics as $C_{ij} = \infty$.

The results of the shortest–path calculations are stored in the source–node's routing table. If the network uses *source–routing*, the entire paths are stored since they must be included in the packets. If the network uses *hop–by–hop* routing, the source node will only need to store the identity of the next–hop node for all destinations.

2.1 Bellman–Ford's algorithm

The Bellman–Ford algorithm can be described as follows:

Definitions: s = Source node. D_j^h = The cost of shortest path from s to node j under the condition, that the path contains at most h links.

Initialization: $\forall h : D_s^h = 0$ and $\forall j \neq s : D_j^0 = \infty$

Iteration: $\forall j : D_j^{h+1} = \min_k \left(D_k^h + C_{kj} \right)$. Keep repeating the iteration until $\forall j : D_j^{h+1} = D_j^h$

Consider the graph in figure 2.
If we want to calculate the shortest paths from node A to all other node in this network we get the following table:
Since $D_j^4 = D_j^3$ for all j, the algorithm has converged. The routing table for node A will then be:

At this point, the shortest–paths from the source node to all other nodes in the network have been found, but in general, we need the shortest–paths between any pair of nodes in the network. For this, it is necessary to execute the Bellman–Ford algorithm N times, where using (in turn) every node in the network as the source node.

[3]Note that the metric and the link's error rate are not directly proportional in this case.

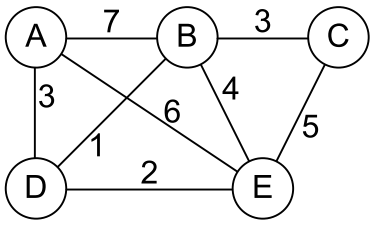

Figure 2: Example network.

	Node B		Node C		Node D		Node E	
h	D_B^h	Path	D_C^h	Path	D_D^h	Path	D_E^h	Path
0 (Init)	∞	—	∞	—	∞	—	∞	—
1	7	A–B	∞	—	3	A–D	6	A–E
2	4	A–D–B	10	A–B–C	3	A–D	5	A–D–E
3	4	A–D–B	7	A–D–B–C	3	A–D	5	A–D–E
4	4	A–D–B	7	A–D–B–C	3	A–D	5	A–D–E

Table 1: Bellman-Ford calculation of shortest paths

Destination	Path (source–routing)	Next–hop (hop–by–hop routing)
B	A–D–B	D
C	A–D–B–C	D
D	A–D	D
E	A–D–E	D

Table 2: Contents of routing table for source node.

2.2 Distributed Bellman–Ford

The Bellman–Ford algorithm can also be implemented in a distributed fashion. To implement this, a node must from time to time transmit information from its entire routing table to all its neighbor nodes. It can be shown that as long as all nodes from time to time transmit this information, the optimal shortest–paths will be determined eventually.

The idea behind the distributed Bellman–Ford is that, whenever a node receives information from a neighbor node, it will determine whether the information will lead to a better (smaller cost) path than previously known. As an example, consider figure 3: Suppose that node A receives the following information from node B: "I can reach node Z with a cost of α" and from C: "I can reach node Z with a cost of β". Node A would then determine, which of these possible ways of reaching node Z would be better by determining the minimum of $C_{AB} + \alpha$ and $C_{AC} + \beta$. If $C_{AB} + \alpha < C_{AC} + \beta$ then the (currently) best path from A to Z is via node B; otherwise it is via node C.

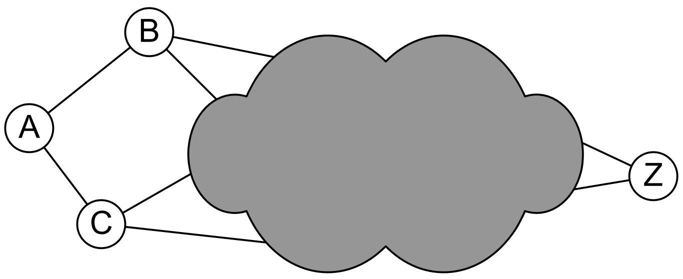

Figure 3: Example network for the distributed Bellman–Ford.

2.3 Dijktra's algorithm

Dijkstra's algorithm can be described as follows:

Definitions: s = source node. D_j = (estimate) of the cost of the shortest path from s to (destination) node j. N = set of all nodes in the network. M = set of nodes.

Step 1: Initialize the set M with just the source node, $M = \{s\}$

Step 2: Find the node with the smallest D_j that is not yet included in the set M and add this node to M: $D_k = \min_{j \notin M} D_j$. $M \leftarrow M \cup \{k\}$

Step 3: For all nodes not (yet) included in M, check if a path through the just–added node (node k) would be better than the current path. $D_j = min(D_j, D_k + C_{kj})$

Step 4: If $N \neq M$ continue from step 2; otherwise the algorithm has terminated.

If we use Dijkstra's algorithm on the network in figure 2, we get table 3. The routing table of node A will of course be the same as in the Bellman–Ford example.

	Node B		Node C		Node D		Node E	
M	D_B	Path	D_C	Path	D_D	Path	D_E	Path
$\{A\}$	7	A–B	∞	—	3	A–D	6	A–E
$\{A, D\}$	4	A–D–B	∞	—	3	A–D	5	A–D–E
$\{A, B, D\}$	4	A–D–B	7	A–D–B–C	3	A–D	5	A–D–E
$\{A, B, D, E\}$	4	A–D–B	7	A–D–B–C	3	A–D	5	A–D–E
$\{A, B, C, D, E\}$	4	A–D–B	7	A–D–B–C	3	A–D	5	A–D–E

Table 3: Dijkstra calculation of shortest paths

3 Quality–of–Service Routing

3.1 Quality–of–Service concept

The concept of *Quality–of–Service* (QoS) has received a lot of attention in recent years with increased interest in distributed multimedia applications. These applications require that the flow of information between the application participants receives some kind of special treatment inside the network. Examples include:

- *Interactive applications* – In this case, the total delay experienced inside the network must be below some upper bound.
- *Streaming applications* – In this case, the flow must receive a minimum throughput, and also required an upper bound on the delay jitter, to minimize the buffer requirements in the end–nodes.

In this lecture note, we will use the following definitions, found in [1]:

Quality–of–Service (QoS): A set of service requirements to be met by the network while transporting a flow.
QoS-based routing: A routing mechanism under which paths for flows are determined based on some knowledge of resource availability in the network as well as the QoS requirement of flows.

Note that the previous definitions were related to routing in the Internet, but we can here generalize to any kind of network.

3.2 More on metrics

The shortest–paths algorithms discussed in the previous section were based on a graph representation of the real network. This graph description is also in QoS–routing, but with one addition. In the previous case, an edge in the graph was associated with *one* (additive) metric. In discussion of QoS–routing in general, an edge is now associated with *multiple* metrics. As before, a metric may be additive, but in addition, in QoS–routing, we may also encounter metrics that are *non–additive.* A typical example is the residual capacity off a link. For instance, consider figure 4. Suppose that the residual capacity on the link A–B is 10 Mbps, while the residual capacity on the link B–C is 6 Mbps. The throughput from A to C is definitely not 16 Mbps, but only 6 Mbps, i.e. the minimum of the residual capacities of the links in the path.

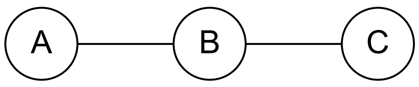

Figure 4: Path with different available bandwidths.

Another example of a non–additive metric is *policies*, i.e. the network operator can impose policies on the links in the network, so that for a given flow, only a subset of all the links may be used.

3.3 QoS–routing categories

The QoS–requirements of the applications can be translated into two types of constraints:

- *Path–constraints* – which are constraints on an entire path through the network, but does not specify any constraints on the individual links that are part of the path. An example of a path–constraint is a requirement by the application that the end–to–end delay must be smaller than some value, D. In this example, the individual link–delays are not important, just as long as the sum of all the link–delays are below D

- *Link–constraints* – which are constraints that every link in a path must satisfy. An example is the residual capacity of a link. If an application wishes to transfer a flow of information with a minimum bitrate of R bps, every link in the path from the source node to the destination node must have an residual capacity of at least R bps.

Given a number of constraints, the purpose of a QoS routing algorithm is then to find an optimal path through the network. In general, there might be a number of paths that all satisfies all constraints – these are called *feasible paths* – so the QoS routing algorithm must select one of these based on some other criteria.

As discussed earlier, metrics can be either additive or non–additive. The second category is simplest one to handle for a QoS routing algorithm, since links that do not satisfy a specific constraint are deleted from the graph as the first step. As an example, suppose that an application requires R bps for its information flow. In that case, all links where the residual capacity is less than R are deleted from the graph, since they would not be part of the final path anyway.

When constraints related to non–additive metrics have been satisfied, the goal of a QoS routing algorithm is to find paths that satisfies the constraints related to the additive metrics. At this point, if there is only one constraint left, the problem can be solved with any of the previously discussed shortest–paths algorithms.

However, if there are multiple constraints left, finding the optimal path is unfortunately NP-complete.

4 QoS Multicast Routing

General discussion about the issues concerning QoS–routing in multicast applications

5 QoS Routing in Ad-hoc Networks

Discussion about the issues of QoS routing in ad-hoc network, especially concerning the imperfect state information.

6 QoS Support in the Internet

Perhaps this section will be included – perhaps not. The original intention was to give a very basic description of IntServ (+RSVP) and DiffServ – but we should perhaps try to find some other material for this. At least there is a reasonably good paper by Lixia Zhang (?) on RSVP.

7 Conclusion

What conclusions?

References

[1] Crawley, E. et al.: "A Framework for QoS-based Routing in the Internet", *RFC 2386*, August 1998

[2] Chen, Shigang and Nahrstedt, Klara: "An Overview of Quality of Service Routing for Next–Generation High–Speed Networks: Problems and Solutions", *IEEE Network*, November/December 1998