HOW TO DISCOVER OPTIMAL ROUTES IN WIRELESS MULTIHOP NETWORKS

Michael Gerharz, Christian de Waal, and Peter Martini
University of Bonn, Roemerstr. 164, 53117 Bonn, Germany
{gerharz, dewaal, martini}@cs.uni-bonn.de

Abstract In this paper, we introduce a distributed algorithm that is able to discover optimal routes in mobile wireless multihop networks using reactive routing protocols. The algorithm is based on Dijkstra's shortest-path algorithm and maps the quality of a path to a delay of the corresponding route request to allow high-quality paths to surpass low-quality paths. With a proper selection of the delay mapping, this approach yields a low overhead and interoperable integration of maximisable routing metrics into existing protocols like AODV and DSR while keeping the route setup delay at a moderate level.

Keywords: Ad hoc networks, Routing, Quality-of-Service, Dijkstra

1. Introduction

Reactive routing protocols provide a ressource efficient solution to the routing challenge in highly dynamic network topologies by discovering a route only when it is actually needed. The source node floods across the network a route request (RREQ) which is unicast back as a route reply from the destination to the source along the discovered route.

In the recent past, much effort has been spent on integrating all kinds of different routing metrics into reactive routing protocols. Motivations come e.g. from QoS-considerations (e.g. maximising the route reliability or the bottleneck capacity) as well as power aware protocols (e.g. minimising the sender-receiver distance in terms of a distance metrics such as energy consumption or number of weak links).

By default, common reactive routing protocols like AODV and DSR [Perkins, 2001] do not support sophisticated metrics, because they process only the first arriving RREQ. While DSR provides limited support by replying to multiple

This work was supported in part by NOKIA and the German Federal Ministry of Education & Research (BMBF) as part of the IPonAir project.

RREQs, this approach is still not able to discover optimal routes, but merely selects the best of several short-delay paths. There are several approaches to incorporate optimal route discovery into reactive routing protocols. However, these approaches focus on special metrics or suffer from high overhead.

In this paper, we will present a generic algorithm applicable to a wide range of diverse routing metrics with very low overhead. The main idea is to use a distributed version of Dijkstra's shortest path algorithm. The key is to schedule the transmission of RREQs in an order that is equivalent to the treatment of the stations in Dijkstra's algorithm.

There are several requirements to an algorithm supposed to discover optimal routes according to some routing metric. Firstly, the *overhead* of the route discovery should be as low as possible compared to conventional reactive routing protocols. Two competing goals are to minimise the number of messages sent over the medium and to minimise the route setup delay. Our algorithm, just like conventional reactive routing protocols, requires exactly one broadcast per station while adding a slight delay to distinguish paths of different quality.

Secondly, the route discovery process should be *interoperable* to the plain routing protocol. This allows for gradual deployment and enables different devices to stress different requirements on the discovered route (whether this is reasonable depends on the scenario). Our approach is fully interoperable with AODV and DSR, but the caching strategy of the latter protocol has to be chosen with care in order not to base routing decisions on outdated information.

The rest of this paper is structured as follows. Section 2 provides preliminaries for the discussions to follow. In section 3, we present related work. Section 4 describes our generic approach to optimal routing and outlines several design choices. After discussing implementational aspects in section 5, we draw conclusions and outline perspectives for further research in section 6.

2. Shortest Path Algorithms & Routing Metrics

A thorough discussion of maximisable routing metrics may be found in [Gouda and Schneider, 2003]. The authors define a routing metric as a 5-tuple $(M, W, \textsc{Met}, p_0, \prec)$ where M is the set of all possible metric values, W is the set of possible edge weights, $\textsc{Met} : M \times W \rightarrow M$ is a metric function which calculates metric values cumulatively, $p_0 = \max(M)$ is used as the initial path metric of a route discovery, and $\prec$ is a less-than total order relation over M so that the routing metric selects the paths of maximal metric values. Sometimes, we will use $\textsc{Met}(c_1, \ldots, c_i)$ as an abbreviation for $\textsc{Met}(\textsc{Met}(\ldots(\textsc{Met}(p_0, c_1), \ldots), c_i)$.

According to this we define a *distance metric* as the 5-tuple $(\mathbb{R}, W, +, 0, >)$. A *reliability metric* is given by $(M = \{x \in \mathbb{Q} | 0 \leq x \leq 1\}, W = M, \cdot, 1, <)$. A *flow metric* is defined as $(M \subset \mathbb{N}, W = M, \min, \max(M), <)$. Defining a maximal weight for the source station has technical reasons as required in section 4. Interestingly, this requirement may be relaxed for a practical implementation (cf. section 5). Anyway, the maximally recordable weight will be limited by a finite value in any real implementation.

Numerous examples for these routing metrics exist in the literature. Power consumption is a frequently used distance metric, flow metrics are commonly used when minimum bandwidth requirements have to be met, and the expected packet loss rate is a typical reliability metric. Further examples for all of these metrics can be found in [Gerharz et al., 2003].

In the following, we will provide a short overview of Dijkstra's single source shortest paths algorithm [Dijkstra, 1959], because it forms the basis for our distributed approach. Consider a network G with a weight function w and a source node s. The shortest paths to all other nodes in the network are basically calculated by the following procedure (cf. [Cormen et al., 1990]):

```
DIJKSTRA($G, w, s$):
  INITIALIZE($G, s$)
  $Q \leftarrow V[G]$
  **while** $Q \neq \emptyset$
    $u \leftarrow$ EXTRACT-MAX($Q$)
      **for** each vertex $v \in Adj[u]$
        **do** RELAX($u, v, w$)
```

The set Q contains all nodes for which the optimal path is not yet known. EXTRACT-MAX selects the node u whose currently best known path is maximal of all nodes in Q. For this node u the maximal metric value is already found. We will denote this maximal value as OPT(u).

The central task of the DIJKSTRA algorithm is the RELAX procedure. It takes three parameters: the two endpoints of a link u, v as well as the weight function w and calculates the metric value $d[v]$:

```
RELAX($u, v, w$) :
  **if** $d[v] \prec$ MET($d[u], w(u, v)$)
    **Then** $d[v] \leftarrow$ MET($d[u], w(u, v)$)
      $nexthop(v) \leftarrow u$
```

The important property of Dijkstra's algorithm in our context is that the RELAX procedure is called exactly once for every edge. This property permits a distributed computation of the algorithm if the distributed calls to RELAX follow an equivalent order as in the centralised case.

3. Existing Distributed Algorithms for Optimal Routing Ad Hoc Networks

A lot of previous work exists on the discovery of optimal routes with reactive routing protocols which basically splits into two groups. The first group tries to keep the route setup delay at a minimum at the price of an increased routing load while the other group favours the opposite.

In DSR, a rudimentary support for optimal routing is provided by the destination which replies to all incoming RREQs. DSR itself uses this approach to discover shortest paths. But, it has also been adopted by other publications, sometimes with slight modifications. In [Tickoo et al., 2003] e.g., the destination does not send multiple route replies, but rather delays the reply in order to answer to the best RREQ just once. This reduces the routing overhead but on the other hand increases the route setup time.

While being very simple, this approach is unable to find the actual optimal route, because the destination is provided with only a subset of all paths. Therefore, extensions were proposed (e.g. [Gupta and Das, 2002], [Bergamo et al., 2004]) to have intermediate stations forward multiple RREQs instead of just the first. While this approach may be able to find the optimal path with low latency, it consumes a huge amount of capacity, because the inherently harmful broadcast storm problem [Tseng et al., 2002] will even be augmented. In contrast, our approach will even lessen the broadcast storm problem.

A different approach is proposed in [Cho and Kim, 2002] and [Chakeres and Belding-Royer, 2003] which is based on the standard AODV discovery scheme. But in contrast to the basic scheme, RREQs are delayed depending on a local state maintained in every station. By this means, the probability for the station to be on the selected route is influenced. Although this approach is specified with weights being assigned to nodes rather than edges, a generic mapping to edge weights is possible due to the fact that only the first arriving RREQ is processed. With that transformation, this approach is merely a special case of ours.

A related approach, also operating on a specific distance metric (power consumption) is described in [Aslam et al., 2003] with algorithm 5, this time using edge weights and assuming a global clock. A station receiving a RREQ delays the forwarding of this RREQ according to the accumulated distance.

In this paper, we will generalise this concept to arbitrary maximisable routing metrics without requiring a global synchronisation of all stations.

4. A Distributed Version of Dijkstra's Shortest Path Algorithm

In this section, we assume that neither the stations nor the medium introduce any further latency other than the one enforced by the algorithm. Furthermore, we assume that the clocks of all stations are synchronised and that without loss of generality the clock starts at 0 for every route discovery. This synchronisation requirement will be relaxed in later sections.

4.1 Key Concepts & Basic Algorithm

The key idea is to make the EXTRACT-MAX procedure implicit by scheduling the broadcast of RREQs distributedly in an equivalent order as the nodes are extracted from the set Q and distribute the computation of RELAX to those nodes receiving the RREQ. In other words, the broadcast time $\text{BT}(u)$ of a RREQ at node u has to fulfil the following condition:

$$\forall u, u' \in G : \text{OPT}(u) \prec \text{OPT}(u') \Rightarrow \text{BT}(u) > \text{BT}(u') \qquad (1)$$

To achieve this, we assign to every path in the network a total RREQ-delay corresponding to the path's cost. Formally, we define a function $D : M \rightarrow \mathbb{R}_0^+$ which is strictly monotonically decreasing. (Note that this is equivalent to finding a mapping of the routing metric to a distance metric.) Having this mapping, a RREQ which is received along a path of value p, is scheduled to be forwarded at global time $D(p)$. Should a better RREQ arrive before $D(p)$ has elapsed, the transmission has to be re-scheduled to the earlier period. Worse RREQs will be discarded. Ties are broken arbitrarily. Formally, we define $\text{BT}(u)$ as the minimal delay of all paths to u. In conjunction with D's monotonicity, it immediately follows that $\text{BT}(u) = D(\text{OPT}(u))$.

This leads to a generic formulation of a distributed version of Dijkstra's algorithm. The set Q is only maintained implicitly and not centrally administered. By broadcasting a RREQ, a node is extracted from Q. Subsequently, the relaxation of an edge is distributed to the broadcasting station's neighbours and triggered by the reception of the RREQ. The RELAX procedure also needs three parameters, however in an accumulated form: the id of the previous hop, the cumulated pathcost p transmitted in the RREQ, and the linkcost c of the last hop:

```
RELAX($u, p, c$):
  if $met \prec \text{MET}(p, c)$
    Then $met \leftarrow \text{MET}(p, c)$
      $nexthop \leftarrow u$
      RESCHEDULE-BT($D(met)$)
```

Under the assumption that no additional delay is introduced by the medium or the stations, this algorithm is able to discover optimal routes which follows immediately from Dijkstra's optimality, because the stations broadcast their RREQ in an equivalent order as the nodes would be extracted from Q. The effect of increasing medium and station latency is out of scope of this paper. However, note that although additional delays lead to suboptimal route assignments this is not necessarily a drawback in practice. By limiting the detour of an optimal route compared to the shortest path, the capacity of the network as well as the energy resources are potentially spared.

As already mentioned in section 3, to discover optimal routes with reactive protocols, a tradeoff has to be found between overhead and route setup delay. Our algorithm does not introduce any overhead in terms of packet transmissions. In terms of byte-overhead, RREQs need to be extended with a *met* field which on the one hand is quite negligible in size and on the other may be spared completely if certain conditions are met (cf. section 5.2). Additionally, note that delaying some of the RREQs stretches the route discovery broadcast storm in time and thereby reduces the peak load on the network.

4.2 Mapping Metric Values to RREQ-Delays

In this section, we will take a look at some example delay mappings, namely linear and logarithmic transformations.

Linear Transformation. In general, a linear transformation of metric values to delays will look like this:

$$D(p) = mp + o \tag{2}$$

where $m > 0$, if $\prec$ actually is a greater-than operator and $m < 0$ otherwise. In general, $o \neq 0$ for $m < 0$ or $\min(M) > 0$. Reasonably, we require o to be chosen such that $D(\max(M)) = 0$ in order to guarantee that optimal paths do not experience any delay at all.

For distance metrics, this leads to delays proportional to the distance (note that this is the special case of algorithm 5 in [Aslam et al., 2003], cf. sec. 3):

$$D(p) = kp \tag{3}$$

where $k \in \mathbb{N}, k > 0$ which we will assume throughout the rest of the paper. For reliability metrics, we have $\prec \equiv <$ and thus $m < 0$. Consequently, with $m = -k$ we define $o = k$ in order to guarantee $D(1) = 0$ which leads to delays proportional to the fragility of the path:

$$D(p) = -kp + k = k(1 - p) \tag{4}$$

Similarly, for flow metrics we define $m = -k$ and $o = k \max(M)$ to get delays proportional to the unused or preoccupied resources:

$$D(p) = k(\max(M) - p) \tag{5}$$

We observe that the maximal delay a RREQ may experience is bounded by k for reliability metrics and by $k \max(M)$ for flow metrics while it is unbounded for distance metrics.

This means, with distance metrics a RREQ may in principle travel through the network arbitrarily long which seems undesirable at first sight. Firstly, RREQs should arrive in a timely manner, because old RREQs will possibly carry outdated information. Furthermore, late RREQs may increase the route setup delay if a better route is not available. However, it may be doubted that this has a great impact in practice. The use of a performance metric in scenarios where a good performance may not be expected in the first place, may be doubted at all. It should be expected that usually a better path is available whose delay is accordingly short.

Logarithmic Transformation. An alternative approach is to use a logarithmic delay transformation which provides underproportional growth of delay for high-quality paths and overproportional growth of delay for lower-quality ones. As an example, we will look at reliability metrics and define:

$$\begin{aligned} D(p) &= k \log p^{-1} &&, p \neq 0 \\ D(p) &= \infty &&, p = 0 \end{aligned} \tag{6}$$

In principle, a logarithmic transformation is also possible for other routing metrics, but care has to be taken to keep the delay positive.

As with a linear transformation, this mapping guarantees a zero delay for 100% reliable paths. But different to linear transformations, 0% reliable paths will be totally discarded (which is a reasonable thing to do). Furthermore, the delay is not bounded but approaches infinity for reliabilities close to zero.

Depending on the number of alternative paths, two pragmatic solutions exist to this problem: unreliable paths below a certain threshold may be totally discarded or delayed by a constant upper limit. The latter approach disregards differences in the reliability of a path and leads to sub-optimal routes while the former approach in effect reduces the connectivity of the network. Which solution is preferable depends on the scenario.

Fig. 1 provides a comparison of linear and logarithmic transformations for reliability metrics. The log-transformation is $D(p) = \frac{3}{10} \log_{10} p^{-1}$ while the linear transformation is $D(p) = \frac{1}{3}p$. Thereby, paths with a reliability of at least 10% will be discovered within 300ms (plus medium and station latency).

Although many more special mappings may be chosen, we refrain from discussing details here.

5. Implementational Aspects

Until now, we have assumed to have the clocks of all stations globally synchronised which is clearly undesirable in real implementations. This may be avoided by computing delays incrementally.

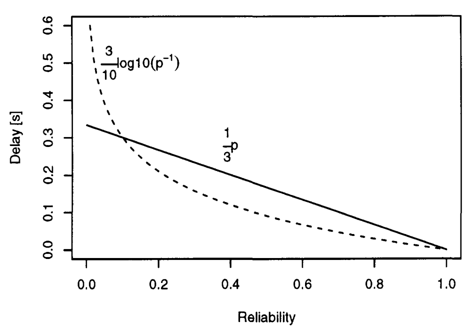

Figure 1. Comparison of linear and logarithmic transformation of reliability metrics

In section 5.1, we describe a straightforward approach that makes use of the path metric value propagated in the RREQ. If this value is implemented as an optional field which is not modified by stations not supporting the extension, this will allow an interoperable integration of routing metrics into existing routing protocols, which is generally desirable. Section 5.2 will show that the message format may even be left unchanged when utilising special delay mappings which allows a fully backwards compatible implementation of routing metrics.

While this approach allows for gradual deployment of novel routing metrics and permits different devices and applications to focus on different performance aspects, having only parts of the stations support a routing metric certainly yields suboptimal routes. Be aware that a partial approach will for some metrics lead to wrong and sometimes even counterproductive decisions (if in particular those stations with high quality links support this and in particular those with bad quality links do not).

5.1 Differential Delay Mapping

In this section, we assume that the pathcost is propagated in the RREQ. With p_{i-1} we denote the metric value contained in the RREQ transmitted on the $i-th$ hop of a path. c_i denotes the linkcost of that link. Additionally, we do not require a global synchronisati on but assume that the deviation of the stations clocks remains in sensible bounds. We formally define:

DEFINITION 1 Differential Delay Mapping

A differential delay mapping *is a function* $d : M \times W \longrightarrow \mathbb{R}_0^+$ *such that for a delay mapping* $D : M \longrightarrow \mathbb{R}_0^+$*:*

$$\forall p \in M, c \in W : D(p) + d(p,c) = D(\textsc{Met}(p,c)) \tag{7}$$

The application of this definition to previously introduced delay mappings provides some interesting insights. For linear delay mappings e.g. we get:

$$\begin{aligned} d(p_{n-1}, c_n) &= (mp_n + o) - (mp_{n-1} + o) \\ &= m(p_n - p_{n-1}) \end{aligned} \tag{8}$$

We observe that the delay calculation comes with very low computational overhead. The procedure requires only two arithmetic operations, because p_n has to be calculated anyway to measure the quality of the path and p_{n-1} is extracted from the RREQ.

Additionally, we notice that d is independent of o. Note that although the local delay is proportional to the absolute difference of the pathcosts, this does not generally imply that d is proportional to the linkcost. This is however true for linear differential mappings of distance metrics which deserves a closer look (recall that $D(p) = kp$, $p_n = \sum_{i1}^{n} c_i$):

$$\begin{aligned} d(p_{n-1}, c_n) &= k(p_n - p_{n-1}) && (9) \\ &= k(\sum_{i=1}^{n} c_i - \sum_{i=1}^{n-1} c_i) \\ &= kc_n && (10) \end{aligned}$$

Obviously, the local delay that a RREQ experiences in a station is independent of the path cost and depends only on the local linkcost which means that it is actually redundant to include the pathcost in the RREQ. This leads us to the notion of local mappings, defined in the following section.

5.2 Local Delay Mapping

In the previous sections, the delay mapping has been calculated from the pathcost. But for distance metrics, it has been shown that this is actually redundant. In this section, we will see that also for other metrics it is possible to make the calculation of the pathcost implicit and to compute the delays of RREQs directly from the linkcosts. Formally, we define:

DEFINITION 2 Local Delay Mapping
A local delay mapping *is a function* $d : W \longrightarrow \mathbb{R}_0^+$ *such that* $(n, m \in \mathbb{N})$*:*

$$\textsc{Met}(c_1, \ldots, c_n) \prec \textsc{Met}(c'_1, \ldots, c'_m) \Rightarrow \sum_{i=1}^{n} d(c_i) > \sum_{i=1}^{m} d(c'_i) \tag{11}$$

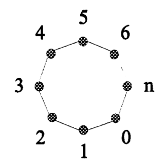

Figure 2 Flow metrics in general are not local

Note that a local delay mapping as defined here is not a special case of a differential delay mapping. Two paths with the same pathcost may arrive at different delays with local mappings whereas they will be guaranteed to experience the same delay with differential mappings by definition.

Reliability metrics may be implemented with a local delay mapping using a logarithmic mapping of reliabilities to delays which we will derive from the differential mapping of a logarithmic transformation (cf. equation 6):

$$\begin{aligned} d(p_{n-1}, c_n) &= \log p_n^{-1} - \log p_{n-1}^{-1} \\ &= \log\left(\prod_{i=1}^{n} c_i^{-1}\right) - \log\left(\prod_{i=1}^{n-1} c_i^{-1}\right) \\ &= \sum_{i=1}^{n} \log\left(c_i^{-1}\right) - \sum_{i=1}^{n-1} \log\left(c_i^{-1}\right) \\ &= \log c_n^{-1} \end{aligned}$$

Since the delay is independent of the pathcost, a local delay mapping for reliability metrics exists via $d(c) = \log c^{-1}$.

For flow metrics, we state the following theorem:

THEOREM 1 *A local delay mapping for flow metrics exists if and only if the size of the network is bounded by a constant N which is known in advance or* $|W| < 3$.

PROOF We will first prove that flow metrics are not local if neither of the two conditions is met by providing a counter-example:

The delay mapping shall impose a lower delay on any path with lower cost, regardless of how much longer this path is. It is easy to see that this is not generally possible: Consider a network of size $n+1, n \in \mathbb{N}$ with circular shape as depicted in figure 2. Let c_i be the linkcost of link $(i, i+1)$ and c_n be the cost of link $(n, 0)$. Furthermore, let $c_n = \min(W)$ and $c_n < c_i < \max(W), 0 \leq i < n$ (recall that $|W| \geq 3$). Then, by definition:

$$\sum_{i=0}^{n-1} d(c_i) < d(c_n) \tag{12}$$

However:

$$\sum_{i=0}^{n-1} d(c_i) > (n-1) \min_{i<n}(d(c_i)) \tag{13}$$

Since $\forall i < n : c_i < \max(W)$, we have $\min_{i<n}(d(c_i)) > 0$. Furthermore, $\min_{i<n}(d(c_i))$ and c_n are constant. This leads to a contradiction to eq. 12, if n is large enough. In general:

$$\exists n \in \mathbb{N} : d(c_n) < (n-1) \min_{i<n}(d(c_i)) < \sum_{i=0}^{n-1} d(c_i) \tag{14}$$

On the other hand, if the network size is bounded by a constant $N \in \mathbb{N}$ which is known in advance, a local delay mapping for flow metrics exists:

$$d(c_n) = kN^{-c_n} \tag{15}$$

Consider a network of size N. Consider a link $(0, N-1)$ of cost c. In the worst case, a path $0, \ldots, N-1$ of length $N-1$ exists that just contains links of linkcost c' only marginally better than c, i.e. $c' = c+1$. Then:

$$\begin{aligned} (N-1) \cdot d(c') &< d(c) \\ \Leftrightarrow N-1 &< d(c)d(c')^{-1} \\ \Leftrightarrow N-1 &< N^{-c}N^{c+1} \\ \Leftrightarrow N-1 &< N \end{aligned}$$

Furthermore, if $|W| = 1$, a valid local delay mapping is trivially defined by $d(c) = 0$. If $|W| = 2$, a valid local delay mapping is defined by $d(c_{min}) = k$ and $d(c_{max}) = 0$. *qed.*

At first sight, it might seem straightforward to simply choose N large enough to meet any imaginable realistic scenario. However, this would require us to increase k as well in order to be able to distinguish also small differences in path quality. But, a large choice of k may result in very large delays even for high quality paths.

6. Conclusions & Further Work

In this paper, we have presented a generic approach to discover optimal routes with reactive routing protocols. The key idea is to delay the forwarding of RREQs according to the pathcost of the discovered path which was used to develop a distributed version of Dijkstra's shortest path algorithm.

This approach does not increase the routing load in terms of packet transmissions and in fact even reduces the peak load during a broadcast storm. On the other hand, the route setup delay is increased. Thus, the delay of RREQs has to be chosen carefully in order to find a good tradeoff between a minimal route setup time and a reliable distinction of high-quality from lower-quality paths. Finally, we have presented a local version of our algorithm which is fully backwards compatible to existing reactive protocols.

Future work will focus on several aspects. First of all, the impact of medium and station latency on a sensible choice of the RREQ delay has to be analysed. Possible improvements may be achieved by using the plain reactive routing protocol to quickly discover *some* route and refine this route selection by additionally running our proposed algorithm. A similar approach would be to limit the maximal RREQ-delay to a relatively short period, but as a compensation forward multiple RREQs if one arriving late yields a significantly better path.

References

[Aslam et al., 2003] Aslam, J., Li, Q., and Rus, D. (2003). A lifetime-optimizing approach to routing messages in ad-hoc networks. In Cheng, X., Huang, X., and Du, D.-Z., editors, *Ad Hoc Wireless Networking*, pages 1–43. Kluwer Academic Publishers.

[Bergamo et al., 2004] Bergamo, P., Giovanardi, A., Travasoni, A., Maniezzo, D., Mazzini, G., and Zorzi, M. (2004). Distributed power control for energy efficient routing in ad hoc networks. *Wireless Networks*, 10(1):29–42.

[Chakeres and Belding-Royer, 2003] Chakeres, I. D. and Belding-Royer, E. M. (2003). Resource biased path selection in heterogeneous mobile networks. Technical Report 2003-18, UCSB, Santa Barbara.

[Cho and Kim, 2002] Cho, W. and Kim, S.-L. (2002). A fully distributed routing algorithm for maximizing lifetime of a wireless ad hoc network. In *Proc. 4th IEEE Conf. on Mobile and Wireless Communication (MWCN 2002).*

[Cormen et al., 1990] Cormen, T. H., Leiserson, C. E., and Rivest, R. L. (1990). *Introdcution to Algorithms*, chapter 25 Single-Source Shortest Paths, pages 514–527. The MIT Press.

[Dijkstra, 1959] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271.

[Gerharz et al., 2003] Gerharz, M., de Waal, C., Martini, P., and James, P. (2003). Strategies for finding stable paths in mobile wireless ad hoc networks. In *Proc. 28th IEEE Conf. on Local Computer Networks (LCN'03)*, Bonn, Germany.

[Gouda and Schneider, 2003] Gouda, M. G. and Schneider, M. (2003). Maximizable routing metrics. *IEEE/ACM Transactions on Networking*, 11(4):663–675.

[Gupta and Das, 2002] Gupta, N. and Das, S. R. (2002). Energy-aware on-demand routing for mobile ad hoc networks. In *Proc. 4th Intl. Workshop on Distributed Computing (IWDC 2002)*, pages 164–173.

[Perkins, 2001] Perkins, C., editor (2001). *Ad Hoc Networking*. Addison-Wesley.

[Tickoo et al., 2003] Tickoo, O., Raghunath, S., and Kalyanaraman, S. (2003). Route fragility: A novel metric for route selection in mobile ad hoc networks. In *Proc. 11th IEEE Intl. Conf. on Networks (ICON 2003).*

[Tseng et al., 2002] Tseng, Y.-C., Ni, S.-Y., Chen, Y.-S., and Sheu, J.-P. (2002). The broadcast storm problem in a mobile ad hoc network. *Wireless Networks*, 8(2/3):153–167.