On Optimal TTL Sequence-Based Route Discovery in MANETs

Dimitrios Koutsonikolas Saumitra M. Das Himabindu Pucha Y. Charlie Hu
School of Electrical and Computer Engineering
Center for Wireless Systems and Applications
Purdue University
West Lafayette, IN 47907
{dkoutson, smdas, hpucha, ychu}@purdue.edu

Abstract

In on-demand multi-hop routing protocols for MANETs such as DSR and AODV, a fundamental requirement for peer-to-peer connectivity is to discover routes to a remote node via flooding of route request messages. Historically, such floodings of requests have used a TTL (time-to-live) large enough to reach all nodes in the network to ensure successful route discovery in one round of flooding. Recently [1], it was shown that the generic minimal cost flooding search problem can be solved via a sequence of floodings with an optimally chosen set of TTLs. The theoretical result, when applied to DSR route discovery, does not take into account optimizations such as route caching and overhearing, which can significantly reduce the frequency and the propagation range of route discovery operations. Equally importantly, the impact of using a sequence of floodings on the packet delivery delay is not clear. In this paper, we study the impact of using the optimal TTL sequence-based route discovery on DSR routing performance. Our results show when caching and overhearing are considered, the route discovery enhanced by an optimal TTL sequence has very similar overhead but higher delay than the basic route discovery mechanism.

1 Introduction

Mobile ad hoc networks (MANETs) are collections of mobile nodes equipped with wireless devices typically deployed in areas where no infrastructure exists. Mobile nodes in a MANET form a network using each other as routers. To support peer-to-peer communication in such networks, *route discovery* (discovering routes to a destination node) is a fundamental requirement. The efficiency of the route discovery mechanism critically affects the scalability of the routing protocol used. For example, in the absence of location information, a route discovery mechanism involving expensive network-wide flooding is typically used by reactive routing protocols. On the other hand, routing protocols that use location information (such as GPS) [8] can use a variety of scalable location services [3] to avoid costly flooding-based route discoveries.

Several non-location-based reactive routing protocols for MANETs, such as DSR [7] and AODV [10], use flooding of ROUTE REQUEST messages to discover routes. A ROUTE REQUEST packet is flooded across the network in a controlled manner (each node broadcasts it only once). The request is answered by a ROUTE REPLY packet either from the destination node or an intermediate node that has a cached route to the destination. The ROUTE REPLY either contains the whole route to the destination, as in DSR, or sets up a path through the intermediate nodes as it travels back to the source, as in AODV. ROUTE REQUEST messages typically contain a TTL value which specifies the number of times a particular ROUTE REQUEST message may be re-broadcast. Historically, reactive routing protocols have used a single flooding with a TTL value larger than the diameter of the network to ensure the discovery of a route with a single flooding which also leads to a low delay. However, such a single flooding can result in high overhead. For example, if the destination D is only 5 hops away, the ROUTE REQUEST messages may be transmitted till up to D hops away where D is the diameter of the network. If we denote N_d^S as the number of nodes d hops away from the source S, then approximately, $N_D^S - N_5^S$ transmissions of ROUTE REQUEST messages are unnecessary.

To reduce the cost of the flooding-based route discovery mechanism, routing protocols can incorporate an expanding ring search (ERS) scheme. In such a scheme, the routing protocol performs several floodings with increasing TTL values, instead of a single network-wide flood. For example, the work in [9] proposes an expanding ring search mechanism for AODV. More recently, it was shown in [1] that the minimal cost flooding search problem using ERS

In *Proceedings of the 2nd International Workshop on Wireless Ad Hoc Networking (WWAN 2005), in conjunction with the 25th IEEE International Conference on Distributed Computing Systems (ICDCS-2005), Columbus, OH, June 6-9, 2005.*

can be solved using an optimally chosen TTL sequence. In this paper, we study the impact of applying this optimal TTL sequence to the route discovery mechanism of DSR.

The tradeoffs between this optimal TTL sequence-based route discovery and the single flooding-based route discovery mechanism are not clear for the following three reasons. First, the theoretical result in [1] assumes no caching or overhearing of routes, which can significantly reduce the frequency and the propagation range of route discovery operations. Second, the use of route discoveries with a larger TTL value spreads routing information to a larger set of nodes, which in turn improves the effectiveness of caching of routes in the network. There exists a tradeoff between the propagation range of route discoveries and the effectiveness of caching. Third, the implication of using the optimal TTL sequence on the delay in finding a valid route is not clear. Such delay is an important performance metric as it directly affects the end-to-end delay of packet delivery. To study these performance tradeoffs, we perform a comparison study of the two route discovery mechanisms via extensive simulations.

Our results show that although theoretically the optimal TTL sequence can minimize the cost of route discovery, when practical optimizations such as caching and overhearing in MANETs are considered, the route discovery enhanced by an optimal TTL sequence has very similar overhead but higher delay than the basic route discovery mechanism.

The rest of the paper is organized as follows. Section 2 briefly reviews the optimal TTL sequence-based flooding search [1] and describes the route discovery mechanism in the original DSR and in the new optimal TTL sequence-based route discovery. Section 3 describes the experimental methodology. Section 4 presents the experimental results. Section 5 discusses the related work. Finally, Section 6 concludes the paper.

2 Background

We first summarize the main results on the optimal TTL sequence-based flooding search [1] that is related to route discovery in DSR. We then describe the original route discovery scheme used in DSR as well as in DSR_TTL which uses the optimal TTL sequence for route discovery.

2.1 Optimal TTL sequence-based search

In [1], the problem of using flooding-based search for a node in a network is studied, and an optimal TTL sequence-based flooding search is proposed. Note that the flooding search problem in a network is conceptually equivalent to the route discovery problem in a MANET. Under this scheme, a node initiates a query packet (analogous to a ROUTE REQUEST) with a preset TTL value and propagates it through the network. When a node different than the desired destination receives the packet, it decrements the TTL value and rebroadcasts it. This process continues until the destination is located or the TTL value reaches zero. In the latter case, a new query is transmitted by the source with a larger TTL value and the whole process is repeated. The problem is to find the optimal sequence of TTL values that minimizes the total transmission cost.

To solve this problem, two different techniques are proposed. First, when the probability distribution of the location of the destination node is known a priori, a dynamic programming formulation is derived to provide the optimal search strategy in terms of the expected search cost. Second, when the probability distribution is not known a priori, randomized strategies are used in order to minimize the worst case search cost. In practice, we typically do not know the probability distribution a priori. This is especially true in MANETs with random mobility. We thus concentrate on the second technique in this paper.

A randomized search strategy is a TTL sequence that consists of random TTL variables instead of deterministic TTL values. It can be shown that given any nonrandom TTL sequence g, we can find at least one new randomized sequence $\hat{g}$, that achieves a lower worst-case cost than that achieved by the nonrandom sequence g. The optimal search strategy among all the random and nonrandom strategies depends on the cost function being used. For two-dimensional networks, a quadratic cost function of the form $C_k = a \cdot k^2$ is a realistic choice. In this case it can be shown that the optimal search strategy $\hat{g} = [\hat{g}_1, \hat{g}_2, ...]$ is created by the nonrandom sequence $g = \{g_k\}$, $g_k = \lfloor r^{k-1} \rfloor$, for $r = \sqrt{2} + 1 \approx 2.4142$, by assigning the following probability distribution to each TTL random variable $\hat{g}_k$:

$$Pr(\hat{g}_k = l) = \begin{cases} \frac{2l+1}{g_{k+1}^2 - g_k^2} & if\ g_k \le l \le g_{k+1} - 1 \\ 0 & otherwise \end{cases} \quad (1)$$

This strategy, as any other randomized one, creates a sequence of sets of TTL values, instead of a sequence of TTL values. After each unsuccessful query, a new TTL value is randomly selected in the next set. The first four sets and the probability for each TTL value for the optimal search strategy under a quadratic cost function are shown in Table 1. This search strategy can be used as the sequence of TTL values for route discovery in MANETs.

2.2 DSR Route Discovery

DSR [7] is a representative multi-hop routing protocol for MANETs. DSR employs a route discovery procedure by which a source node discovers a route to a destination for which it does not already have a route in its cache. The process broadcasts a ROUTE REQUEST packet which is flooded

Table 1. The first 4 sets of TTL values and their probabilities of being chosen in the optimal search strategy under quadratic cost function

Seq.	Possible TTL values	Corresponding Probabilities
1	1	1
2	2, 3, 4	5/21, 7/21, 9/21
3	5, 6, ..., 13	11/171, 13/171, ..., 27/171
4	14, 15, ..., 32	29/893, 31/893, ..., 65/893

across the network in a controlled manner. In addition to the address of the original initiator of the request and the target of the request, each ROUTE REQUEST packet contains a route record, which records the sequence of hops taken by the ROUTE REQUEST packet as it propagates through the network. ROUTE REQUEST packets use sequence numbers to prevent duplication. The request is answered by a ROUTE REPLY packet either from the destination node or an intermediate node that has a cached route to the destination.

DSR also employs a route maintenance procedure which monitors the operation of the route and informs the sender of any routing errors. If a route breaks due to a link failure, the detecting host sends a ROUTE ERROR packet to the source which, upon receiving it, removes all routes in its cache that use the hop in error.

TTL of Route Discoveries When a node initiates a ROUTE REQUEST packet, it first sends the packet with TTL=1. It waits for a period of *base_delay* for any reply to arrive. If no reply arrives in this time, it floods the whole network and waits for a backoff period $T_{backoff}$. If it still receives no reply after the backoff period, it repeats the two-step flooding, but this time with the backoff period increased to $2 \cdot T_{backoff}$. The process repeats until the destination is located or the maximum number of retries is reached.

Optimizations To reduce the cost of the route discovery, each node maintains a cache of source routes that have been learned or overheard, which it uses aggressively to limit the frequency and propagation of ROUTE REQUESTS. In addition, both route discovery and maintenance benefit from optimizations such as overhearing routes and route errors made possible by the broadcast nature of the medium access environment.

2.3 DSR_TTL Route Discovery

DSR_TTL is a modified version of DSR which incorporates an optimal TTL sequence-based route discovery mechanism. In DSR_TTL, after each unsuccessful ROUTE REQUEST, a new ROUTE REQUEST is initiated with a TTL chosen according to the optimal sequence as described in Section 2.1. Accordingly, the waiting period in each ROUTE REQUEST is adjusted to be proportional to the TTL value used in each ROUTE REQUESTS, i.e.,

$$waiting\_period(ttl) = ttl \times base\_delay \qquad (2)$$

Given the current TTL value in use, this equation is used to estimate the waiting period in DSR_TTL. We use a *base_delay* of 30 msec (same as in the original DSR implementation) unless otherwise specified.

3 Experimental Methodology

Our simulations are performed using the Glomosim [12] simulator. We consider a network of 100 nodes in an area of 1500m x 1500m. We also consider a network of 1000 nodes with the same density as the 100-node scenario. The mobility scenarios are generated using a modified "random waypoint" model [11], in which nodes move at a speed uniformly distributed between 1-19 m/s. A wireless radio with 2 Mbps bit rate and 250m transmission range is used. The simulation duration chosen is 1200s. The simulation results are averaged over 10 runs. The communication pattern has Constant-Bit-Rate (CBR) traffic sources with a packet size of 512 bytes. Details of the specific communication pattern used for each experiment are provided with the corresponding results. In all simulations, we made sure that the network is connected at all times, i.e., there is no network partition.

The following metrics are evaluated for the routing protocols: (1) *Routing overhead* – The number of control packets transmitted, with each hop-wise transmission of a control packet counted as one transmission; (2) *Packet delivery ratio* (PDR) – The ratio of the data packets delivered to the destinations to those generated by the CBR sources; (3) *Average delay* – The average end-to-end delay of data packet delivery, including all possible delays caused by buffering during route discovery latency, queuing at the interface queue, retransmission delays at the MAC, and propagation and transfer times; and (4) *Request latency* – The average delay in discovering a new route to a destination, i.e., the time elapsed since the initiation of a route discovery to the time the actual route is discovered.

4 Experimental Results

To verify the theoretical results in [1], we first consider a single source and simulate a static network with 100 nodes with optimizations such as route caching or overhearing disabled. We then enable caching and overhearing to study their impact. Further, we increase the network size to 1000 nodes, as the theoretical results in [1] assume infinitely large networks. Finally, we consider a mobile network with multiple sources.

Table 2. Comparison between DSR and DSR_TTL for a small network without caching.

	DSR	DSR_TTL
Discoveries	99	99
Floodings	96	192
Total Overhead	13084	11715
Packet Delay	0.1345	0.2358
PDR	100	100

4.1 Static Networks with a Single Source

4.1.1 A Small Network

In this experiment, we use a static network with 100 nodes uniformly placed in a square terrain of area 2.25 km^2. The traffic pattern consists of a single source sending exactly one packet to every other node in successive time intervals of 10 msec.

Without caching In this case, nodes do not cache any routing information or overhear packet transmissions. Thus, the source has to perform a route discovery each time it wishes to send a packet to a destination and only the destination sends a reply for this request. Hence, in case of 100 nodes, 99 route discoveries are initiated by the source, one for each of the other 99 nodes.

Table 2 shows the measured results. The PDR in both protocols is 100% since the packet rate is low and no mobility exists. The results show that DSR_TTL always initiates more floodings than DSR. A flooding is a ROUTE REQUEST with a TTL > 1. In DSR, a flooding uses the maximum TTL value to make sure that the ROUTE REQUEST will be propagated through the whole network. Since the network graph is always connected, a flooding always locates the destination. However, this is not true in DSR_TTL, which can involve multiple floodings for a single route discovery. In this case, a flooding may use a TTL value larger than 1, and the TTL is increased after each unsuccessful ROUTE REQUEST. In other words, the first flooding, which uses a TTL value from the second set of TTL values (2, 3 or 4), may not always reach the destination, and additional floodings with larger TTL values will often be required. For this experiment, we observe an increase in the number of floodings of about 100%. This increase is expected to be larger for larger network sizes. In spite of more floodings, DSR_TTL incurs 12% lower total overhead than DSR.

Table 2 also shows that the packet delay in DSR_TTL is about 1.75 times the delay in DSR. In DSR_TTL, the same mechanism that reduces the total overhead is responsible for large delay, since the source has to initiate successive floodings for most of the destinations and wait for possible replies after each one. In contrast, in DSR, one flooding is typically enough to locate the destination.

A critical factor that affects the total packet delay is *base_delay*. Note that the total packet delay can be divided into two parts: the delay in obtaining a route to the destination (request latency) and the delay for the packet to reach the destination. The value of *base_delay* only affects request latency. *base_delay* determines the *waiting_period* which in turn determines the amount of time a node waits for a reply to a specific flooding. Ideally, after initiating a flooding with TTL=k, a node S should estimate its *waiting_period* such that it waits only till all N_k^S nodes have received the ROUTE REQUEST and a ROUTE REPLY from the furthest nodes can reach back to S.

If *base_delay* is very small, the waiting period may expire before the ROUTE REQUEST reaches the furthest nodes. In that case the source may unnecessarily initiate a new flooding with a larger TTL value. On the other hand, if the *base_delay* is very large, the node will unnecessarily wait before initiating the next flooding. Note that *base_delay* affects DSR_TTL more than DSR, since DSR uses it only once when TTL=1, whereas DSR_TTL uses it after each unsuccessful ROUTE REQUEST. In order to find the optimal *base_delay* and reduce the total delay, we ran simulations varying the *base_delay*. Figures 1–3 show the request latency, the ratio of request latency, and the total overhead, respectively, for the two protocols as *base_delay* is varied. The ratio of request latency is defined as $\frac{request\ latency\ in\ DSR\_TTL}{request\ latency\ in\ DSR}$.

The following observation can be made for Figures 1–3. First, they confirm that *base_delay* affects DSR_TTL much more than DSR. Second, they show that there is a tradeoff between the request latency (and subsequently the total delay) and the total overhead. The total overhead remains constant for a large range of *base_delay* values, but it starts to increase when *base_delay* decreases below a certain threshold, and this behavior is more pronounced in DSR_TTL. The overhead increases because unnecessary floodings with larger TTLs may be initiated as the source does not wait to ascertain whether the destination was reached using the current TTL. Thus, we cannot infinitely decrease *base_delay* to reduce total delay. Figure 3 shows that the threshold below which the overhead starts increasing is 15 msec. Thus we selected *base_delay* = 15 msec as the optimal *base_delay* in the rest of the experiments.

We repeated the simulations of the same scenario, using the optimal *base_delay* of 15 msec. The results are shown in Table 3. As we can observe, using the optimal *base_delay* reduces the total packet delay in both protocols and reduces the delay ratio, defined as $\frac{packet\ delay\ in\ DSR\_TTL}{packet\ delay\ in\ DSR}$, to 1.4 from 1.75 when using a *base_delay* of 30 msec, without affecting the total overhead or the number of floodings. However, DSR_TTL still exhibits 40% higher average delay than DSR.

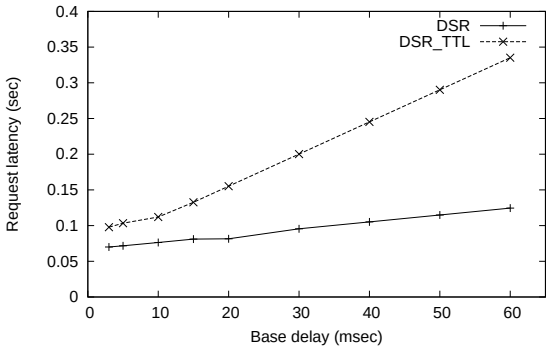

Figure 1. Request latency

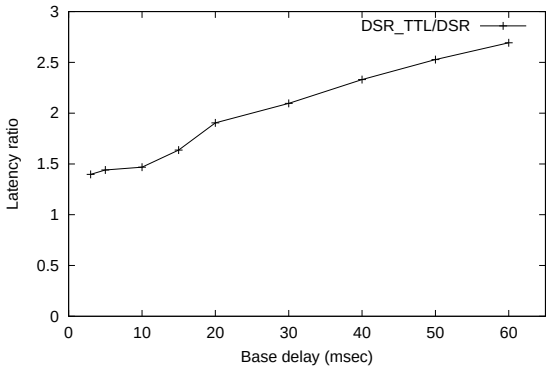

Figure 2. Request latency ratio

With caching In this case, nodes cache all routing information they receive. When a node forwards a ROUTE REQUEST towards the destination or a ROUTE REPLY towards the source, it extracts and stores the route to the source or the destination, respectively, as well as the routes to all intermediate nodes. Moreover, nodes also learn routes by overhearing packets transmitted or received by neighbors. Thus, when a source desires to send a packet, it first looks for a route to the destination in its cache and it only initiates a ROUTE REQUEST if it does not find a route. Additionally, ROUTE REPLIES may come not only from the destination, but also from any intermediate node that knows a route to the destination. In such a case, the intermediate node will not rebroadcast the ROUTE REQUEST packet. This reduces the transmission cost and the total packet delay, since in many cases a ROUTE REQUEST will not need to reach the destination to discover a route. However, the overhead due to replies will increase due to more nodes replying. To investigate how caching affects DSR_TTL, we use the same scenario as before but with caching enabled. Table 4 shows the results.

Compared to Table 3, the PDRs in both protocols remain as 100%. Due to caching, the numbers of route discoveries in DSR and DSR_TTL are reduced to 49 and 56, respectively, which results in the corresponding number of floodings in the two protocols to be reduced by factors of 2.1 and 1.8, respectively. Clearly, DSR benefits more from caching than DSR_TTL. This is expected, as DSR uses more aggressive TTLs which results in more routes being cached in the network. The reduced frequency of route discoveries translates into reduced total overhead – although DSR_TTL still has a lower total overhead than DSR, the difference (3%) is now smaller than without caching (12%). One factor that contributes to the still higher overhead in DSR than in DSR_TTL is as follows. In case of without caching, a single ROUTE REPLY is sent back to the source. In case of caching, multiple ROUTE REPLIES can be sent back to the source by intermediate nodes that have cached routes to the destination. Thus caching can potentially cause increased ROUTE REPLY transmissions. The total delay in DSR increases to 51% larger in DSR_TTL than in DSR, which is 11% higher compared to without caching.

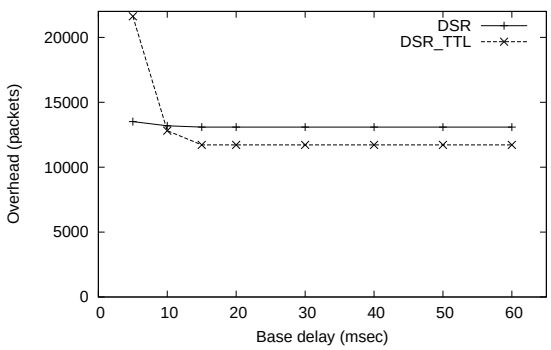

Figure 3. Overhead

Table 3. Comparison between DSR and DSR_TTL using optimal *base_delay*

	DSR	DSR_TTL
Discoveries	99	99
Floodings	96	192
Total Overhead	13084	11715
Packet Delay	0.1200	0.1683
PDR	100	100

4.1.2 A Large Network

DSR_TTL could potentially be more useful in large networks. Hence we investigate the performance tradeoffs between the two protocols in a 1000-node network. As in the 100-node network scenario, there is only one source that sends one packet to every other node in time intervals of 10 msec. Table 5 shows the results.

Table 5 shows that the PDR is still close to 100% in the large network both with and without caching. The total overhead is still lower in DSR_TTL than in DSR, but

Table 4. Comparison between DSR and DSR_TTL for a small network with caching

	DSR	DSR_TTL
Discoveries	49	56
Floodings	46	104
Total Overhead	6542	6366
Delay	0.0635	0.0958
PDR	100	100

Table 5. Comparison between DSR and DSR_TTL for a large network with and without caching.

	without caching		with caching	
	DSR	DSR_TTL	DSR	DSR_TTL
Discoveries	999	999	459.67	504
Floodings	995	3123	456.3	1554.7
Total Ovhd.	1118239	1083311	542008	539233
Delay	0.2762	0.5303	0.1556	0.2863
PDR	99.83	99.93	100	100

the gap in total overhead between the two protocols is now much smaller compared to the 100-node network. Without caching, this gap is about 3% for 1000 nodes, while it was 12% for 100 nodes. With caching enabled, the gap almost disappears; the total overhead in DSR_TTL is only 0.51% less than in DSR. On the other hand, the total delay in DSR_TTL in the large network is 92% and 84% higher than DSR, without and with caching, respectively. Hence we conclude that as the network size increases, DSR gives better overall performance than DSR_TTL.

4.2 Mobile Networks

In this section, we examine the behavior of the two protocols in a mobile scenario typically used in the protocol studies in ad hoc networks (for example, [4]). Most such studies assume 50 or 100 nodes moving with the random waypoint mobility model and CBR traffic sources.

The scenario differs from the static scenario in two aspects. First, instead of sending one packet to each destination node, each source generates CBR traffic to a destination node, and thus route caching will benefit both protocols, as the overhead and delay from discovering a route can be amortized over many subsequent data packets from the same source. In particular, the average delay is expected to be much lower than the static scenario we considered above. Second, the potential benefits of route caching are not unlimited, as nodes are mobile, and cached routes can break.

The specific mobile scenario we consider consists of 100 nodes in a square area terrain of $1km^2$. A pause time of 300 seconds is chosen. Among the 100 nodes, 20 sources are randomly selected and each source sends packets to one another node at a rate of 0.5 packets/second. All the optimizations (caching and overhearing) are enabled for both protocols. Table 6 shows the results.

Table 6. Comparison between DSR and DSR_TTL in a mobile network of small size.

	DSR	DSR_TTL
Discoveries	545	653
Floodings	203	365
Total Overhead	48006	46502
Delay	0.0466	0.0538
PDR	80.58	79.10

The following observations can be made from Table 6. First, the PDR in both protocols is approximately 80%. This is lower than the static scenario as a route between two nodes breaks as the nodes along the route move. Second, although DSR_TTL requires more route discoveries, it still incurs 3% lower overhead than DSR. This is consistent with the results observed in the static 100-node network. Third, the average packet delay for DSR_TTL is now 15% higher than in DSR. This is because the delay for discovering a source route when sending a data packet is amortized among the data packets sent subsequently which simply use the cached routes.

We also simulated a 1000-node mobile network but the combined effects of reduced capacity in large networks [5] as well as the load offered by CBR traffic significantly reduce the packet delivery ratio. We also simulated a 100-node mobile network without caching, but the PDR is again very low due to the significantly increased routing overhead caused by route discovery for every data packet.

4.3 Summary of Results

In summary, with the optimal *base_delay*, in a 100-node static network with one node sending one packet to every other node, DSR_TTL incurs 12% lower total overhead but 40% longer packet delay than DSR without caching and overhearing. With caching and overhearing, the gap in total overhead is reduced to 3% and in packet delay is increased to 51%. Thus, DSR benefits more from caching and overhearing than DSR_TTL. In a 1000-node static network and with caching and overhearing, the gap in total overhead is further reduced to 0.5% but the gap in packet latency is increased to 84%. Thus, as the network size increases, DSR_TTL experiences diminishing benefit in the total overhead and increased packet delay. Finally, in a realistic 100-node mobile environment with CBR packet sources, the gap in total overhead remains as 3%, same as in the 100-node static scenario, but the gap in average packet delay is reduced to 15%, from amortization due to route caching.

5 Related Work

In [9], an expanding ring search scheme for AODV is proposed. In this scheme, a ROUTE REQUEST is initiated with a small TTL value, followed by ROUTE REQUESTS with successively incremented TTL values upon unsuccessful route requests, until a certain threshold is reached at which point a ROUTE REQUEST is flooded across the network if no route has been found. The waiting period is set as $waiting\_period(ttl) = 2 \cdot ttl \cdot node\_traversal\_time$ where *node_traversal_time* approximates the time required by the node to process and transmit a packet. Since this scheme does not use the optimal TTL sequence, it is in theory less optimal than DSR_TTL [1].

Expanding ring search is also examined in [6]. The authors try to find the optimal initiating TTL value, the optimal TTL increment, and the threshold after which a network-wide flooding should be initiated. They conclude that: 1) the use of start and increment TTL values greater than 1 results in reducing both overhead and delay, 2) there are optimal values for these two parameters depending on the network topology, and 3) the threshold is a small value in the range $[2 - 4]$ and it is topology independent.

In [2], the authors compare expanding ring schemes in terms of the overhead and latency. They prove that a two-tier expanding ring search (i.e., a search in two rounds) is always better than a single flooding in terms of the overhead. Another interesting conclusion is that the two-tier scheme used in DSR has the worst performance among all two-tier schemes (although it is still better than flooding the whole network on the first round), while using a TTL value of M/2 in the first round (where M is the longest hop distance) and flooding the network in the second round achieves the best performance among all two-tier schemes.

In addition to modifying the ERS scheme, other techniques have been proposed to make route discovery more efficient such as using the past history of hop distance to decide on the initial TTL value [9] and localization of route discoveries [9].

6 Conclusions

In this paper, we study the impact of using optimal TTL sequence-based route discovery on the performance of DSR. The main conclusion drawn from our study is that although theoretically the optimal TTL sequence can minimize the cost of route discovery, when practical optimizations such as caching and overhearing in MANETs are considered, the route discovery mechanism enhanced by an optimal TTL sequence has similar overhead but higher delay than the basic mechanism. Thus, the choice of which protocol to use depends on the nature of the application. For delay sensitive applications DSR is a better choice whereas in networks with highly energy constrained devices and for delay tolerant applications, DSR_TTL may be a better choice.

Although DSR_TTL reduces the total overhead marginally, it creates more congestion near the sources. Thus if a few nearby sources communicate with faraway nodes, their vicinity will be more congested in DSR_TTL than in DSR.

Although we study the optimal TTL sequence-based route discovery in the context of DSR, similar tradeoffs are likely when such an approach is used in other protocols such as AODV. The one significant difference would be that AODV does not have very aggressive caching mechanisms like DSR which could increase its overhead compared to an AODV version that uses the optimal TTL sequence.

Acknowledgment

This work was supported in part by NSF grant ANI-0338856.

References

[1] N. Chang and M. Liu. Revisiting the TTL-based controlled flooding search: Optimality and randomization. In *Proc. of ACM MobiCom*, September 2004.

[2] Z. Cheng and W. B. Heinzelman. Flooding strategy for target discovery in wireless networks. In *Proc. of ACM MSWiM*, September 2003.

[3] S. M. Das, H. Pucha, and Y. C. Hu. Performance comparison of scalable location services for geographic ad hoc routing. In *Proc. of IEEE INFOCOM*, March 2005.

[4] S. R. Das, C. E. Perkins, and E. M. Royer. Performance comparison of two on-demand routing protocols for ad hoc networks. In *Proc. of IEEE INFOCOM*, March 2000.

[5] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, March 2000.

[6] J. Hassan and S. Jha. On the optimization trade-offs of expanding ring search. *Lecture Notes in Computer Science (LNCS) Springer Verlag*, December 2004.

[7] D. B. Johnson and D. A. Maltz. *Dynamic Source Routing in Ad Hoc Wireless Networks*. Kluwer Academic, 1996.

[8] B. Karp and H. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proc. of ACM MobiCom*, August 2000.

[9] S.-J. Lee, E. M. Belding-Royer, and C. E. Perkins. Scalability study of the ad hoc on-demand distance vector routing protocol. *Int. J. Netw. Manag.*, 13(2), 2003.

[10] C. E. Perkins and E. M. Royer. Ad hoc on-demand distance vector routing. In *Proc. of IEEE WMCSA*, February 1999.

[11] J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. In *Proc. of IEEE INFOCOM*, April 2003.

[12] X. Zeng, R. Bagrodia, and M. Gerla. Glomosim: A library for parallel simulation of large-scale wireless networks. In *Proc. of PADS Workshop*, May 1998.