Roadmap-Based End-to-End Traffic Engineering for Multi-hop Wireless Networks

Mustafa Omer Kilavuz, Ahmet Soran, and Murat Yuksel
Department of Computer Science and Engineering, University of Nevada, Reno, NV 89557
mkilavuz@cse.unr.edu, asoran@cse.unr.edu, yuksem@cse.unr.edu

***Abstract*—End-to-end traffic engineering (E2E TE) in multi-hop wireless networks is hard since the nodes are dynamic and can fail or move. As the network size grows, the dynamism of the nodes prohibits E2E TE approaches using paths as well as network state defined using highly variant nodes and links. Thus, E2E TE over such dynamic networks require the capability of expressing E2E paths and network state in a manner independent of the nodes. We propose a roadmap-based trajectory planning scheme to perform E2E TE over multi-hop wireless networks. We illustrate how our roadmap-based approach can automate the process of planning/selecting the trajectories so that better balancing of the traffic is achieved. We compare our roadmap-based approach to its shortest-path routing counterpart, Greedy Parameter Stateless Routing (GPSR), and show that beneficial tradeoffs can be attained.**

I. Introduction

A network with well-engineered traffic exhibits the property of well balanced traffic on it. Such load balancing is needed for various desired performance measures such as reducing hotspots to attain a higher aggregate network throughput and reducing vulnerability to failures caused by overloading of routers. Particularly for wireless (sensor) networks where power is precious, traffic engineering (TE) increases the lifetime of the network. However, it is not an easy task to engineer/steer the traffic so that an evenly distributed traffic emerges on the network.

Although *local* techniques, such as back-pressure methods [1], can be developed for congestion-based traffic engineering, they tend to be unstable and generate unreliable paths that end up becoming useless for the end-to-end transport protocols like TCP. Because of this drawback, researchers have focused on "end-to-end" traffic engineering (E2E TE) techniques where the paths are decided in a *global* manner considering the source and destination pair for each flow. Such global look at the problem allows the network administrator to resolve collisions among E2E flows and attain better load balancing across the whole network. Backbone ISPs regularly employ such end-to-end traffic engineering techniques using protocols like MPLS [2]. Further, such E2E TE allows accommodation of application-specific requirements as well. For example, the administrator could steer delay-sensitive application traffic on to shorter paths while trying to balance the load.

However, a correspondent E2E TE capability for multi-hop wireless networks has not been achieved. Likewise, there have not been enough studies on E2E TE in multi-hop wireless networks so far. In a multi-hop wireless network, where nodes and links could be dynamic (i.e., failing or moving), it is hard to support application-specific requirements such as maximum throughput, minimum delay and minimum packet loss. On one hand, the transport layer protocols have to be robust and tolerant to packet losses or delays (which requires stable E2E paths or services being provided by the network layer); on the other hand, the network layer protocols must be flexible and capable of "scalably" dealing with the dynamics of the network (which causes variation in the service quality provided to the upper layers). Also, they both need to adapt to the specific needs of the applications.

To steer E2E flows for load balancing and hosting application-specific needs, trajectory-based routing (TBR) [3], [4] is promising routing scheme that enables definition of E2E paths over a network where nodes and links are dynamic and, potentially, moving. In TBR, the E2E path is defined in terms of a trajectory and is not directly dependent on the nodes being traversed. The source node encodes the trajectory into data packets' headers and each intermediate node forwards the data packets to its neighbors according to the trajectory decoded from the packet headers. However, trajectories need to be adjusted to avoid holes or obstacles in the network, which require collection of information about the network so that the source node can pick the E2E trajectories properly.

We propose to use roadmaps [5] to generate obstacle-free trajectories while keeping the nice features of TBR, such as flexibility to enable E2E TE and agility to the application-specific constraints and dynamic nature of multi-hop wireless networks. In robotics, roadmaps are great tools and are widely used for robot navigation. A roadmap is a weighted-graph which is used to find a safe path between a start and a goal point. In [6] and [7], robot navigation is performed by the help of a roadmap and is aided with a wireless sensor network. The network collects information about the environment and the robot builds and adjusts the roadmap according to this information and makes decisions by following a path on the roadmap. Since we work on a mobile network and TBR is not dependent to some specific nodes, a roadmap is suitable for generating trajectories.

Load balancing is easier with a roadmap and trajectories than other methods with heavy calculations, potentially requiring information about the underlying topology. In order to achieve longer network lifetime, for example, we need to prevent the use of specific nodes more often than the others. This is achievable by using different nodes for different trajectories, changing the trajectory after some time for long-lasting connections, and also by dividing the traffic into several trajectories. While a trajectory is in use, we increase the weights of the corresponding roadmap edges over time. The

978-1-4799-4894-9/14/$31.00

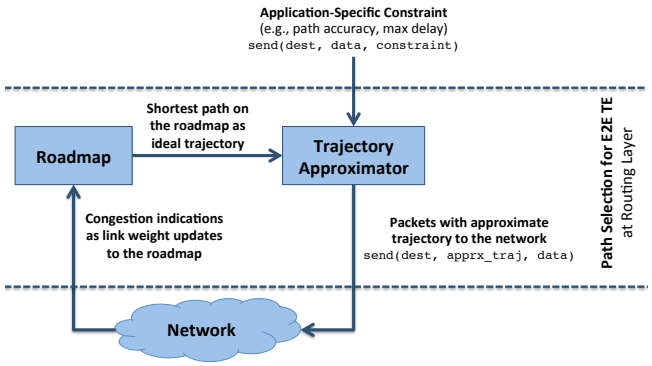


Fig. 1. Path selection for E2E TE.

shortest path is periodically recalculated to see if a shorter path exists. This forces the trajectories to be distributed over the area and balances the overall traffic.

Further, since the obstacle avoidance and E2E TE problems are dependent, we realized they could be solved with a common method. As the baseline, we assume that there is minimum information about the environment; however, to provide better solutions we assume feedback from the lower layers for some methods. We further assume that the wireless nodes are distributed on a planar field and know their positions using Global Positioning System (GPS) [8] or a similar system. The source node gets the position of the destination node from a distributed hash table (DHT) [9], so that it knows which coordinates to send the packets. The application provides constraints and the route selector generates the ideal trajectory and passes it to TBR at the network layer and TBR does its best to deliver the packets to the destination. In this manner, E2E TE in multi-hop wireless networks becomes all about choosing the best obstacle-free trajectories and the parameters for each connection and let the lower layers do their jobs.

The rest of the paper is organized as follows: In Section II, we review the literature for the usage of roadmaps in robotics. Then, in Section III, we describe our proposed framework with roadmaps and E2E TE. Section IV presents an initial simulation-based evaluation of our approach. Section V discusses simulation results and compare both protocols. Finally, Section VI includes conclusion and future work.

II. Related Work

Load balancing in multi-hop wireless networks has been a heavily studied area. Most of the prior work focused on maximizing lifetime of a wireless sensor network with the understanding that sensor nodes have limited power. A plethora of ideas were explored for solving the problem of holes geographic routing techniques [10] and mitigating hotspots via back-pressure methods [1]. Back-pressure techniques suffer from convergence issues and may need prohibitively large overhead to keep the routing under control [11]. Although efforts for scaling back-pressure techniques has been undertaken (e.g., by splitting the network into clustering [12] or by establishing stability via geographic routing [13]), their nature is to handle the globe load balancing problem via local moves which is how back-pressure works.

Some recent work aimed to reach global solutions to the problem by more proactive designs [14] or involving more complex routing constructs like multi-paths [15], multicasts [16], or a mix of multicasts and anycasts [17]. However, none of these techniques allow E2E steering of flows. To the best of our knowledge, our roadmap-based approach is the first to provide a technique to enable source nodes to steer the flows without being explicitly dependent on the network.

In general, roadmaps are heavily used for path planning and summarizing network state. In robotics, for example, roadmaps [5] are widely used for robot navigation. A roadmap is a weighted graph which is used to find a safe path between a start and a goal point in a continuous space. Bhattacharya *et. al.* [6] introduced an approach of building a roadmap with the global information collected from a wireless sensor network. The roadmap is then used by robots to find a safe path without needing to navigate the environment to find such a path. Another work [7] proposed a different approach to the "safe path" problem. Instead of building the roadmap independent of the distribution of the wireless sensor network, they build the roadmap on the network. Some of the nodes are selected as the milestones of the safe path from the start to the goal. Each milestone builds paths to other nearby milestones through intermediate nodes that are called edge nodes. Although our work has aspirations from these robotics literature, it has many differences and has a very different goal: maximize network throughput by balancing the traffic.

III. Roadmap-Based E2E TE Framework

The problem of selecting the best path for balancing the traffic load (E2E TE) in the network is difficult [18]. It further gets complicated if one wants to accommodate application-specific constraints (e.g., path accuracy or max delay) while selecting the paths. In backbone networking, administrators tune IGP link weights to balance the traffic on shortest paths and use MPLS to steer the traffic towards specific (non-shortest) paths. The basic idea of our E2E TE framework is to use roadmaps for deciding which paths to select.

As illustrated in Figure 1, the source node maintains a roadmap, summarizing the network state, and updates it based on congestion indications from the network. When sending data to a destination, the source consults the roadmap and finds the shortest path on the roadmap. It uses this shortest-path as the "ideal trajectory" to follow. It further updates this ideal trajectory to an "approximate trajectory" which is understandable by the network nodes. This approximation helps reduce the forwarding complexity of the packets at the nodes by simplifying the ideal trajectory. It also allows consideration of application-specific factors such as path accuracy (how well the packets follow the ideal trajectory) and max delay. Our earlier work addressed how to balance the tradeoff between application-specific constraints and the routing complexity at the network [19]. In this work, we use path accuracy as the application-specific constraint.

Our framework achieves the traffic load balancing by allowing the link weights on the roadmap to be updated. As congestion indications pertaining to a network region arrive, the weights of the roadmap links corresponding to that region could be increased. Likewise, as a particular network region

stays idle over time, the roadmap links' weights for that region are decreased. This adaptive tuning of the link weights will steer the shortest paths selected from the roadmap to be less utilized ones, and hence attain a balanced traffic load.

A. Forming Roadmaps

The first step of building a roadmap is to generate imaginary vertices randomly located on the region covering the network. Yet, two vertices cannot be closer to each other than a distance d_c and these vertices are connected to each other with an edge if the distance between them is less than d_f. Initially each edge's weight is set to a determined value E_0 which will be changed depending on usage of the roadmap. Source-destination pairs are assumed to be a part of the roadmap. Therefore, shortest path can be calculated over roadmap. The sequence of edges on the shortest path becomes the ideal trajectory for the transmission. Figure 2 shows a sample roadmap and an ideal trajectories between a source node and a destination node. Roadmaps are generated independently and stored locally by each node on the network.

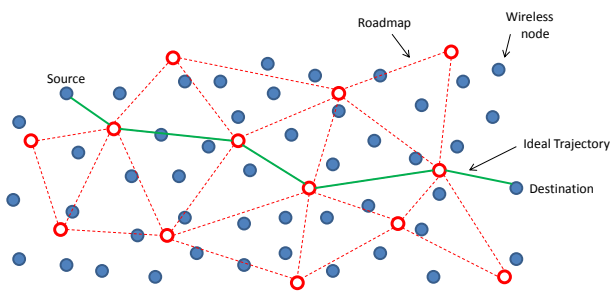


Fig. 2. Roadmap and generated ideal trajectory

B. Representations for Approximate Trajectory

To ease the processing during forwarding, we limit the representation of the approximate trajectory to a small set of curves. The ideal trajectory is expressed as a concatenation of these simpler representations. In [20], we used three polynomial curves to express the approximate trajectories: line, quadratic polynomial, and cubic polynomial. One limitation with polynomial curves is that the curve cannot have more than one point on a specific x-intercept. In order to obtain a curve which has more than one point on an x-intercept, one must generate a similar curve and change its orientation. Bézier curves do not have this limitation because they are defined by control points.

Bézier curves are more practical and flexible compared to polynomial curves and the parameter t is very useful for packet forwarding, because, curves are defined by control points so they can have more than one point on a specific x-intercept. Also, Schneider's nearest point on curve algorithm [21] is used to calculate reference points on curves. We used three Bézier curves for this purpose:

- Linear Bézier curve: $B(t) = (1-t)P_0 + tP_1$, $t \in [0, 1]$
- Quadratic Bézier curve: $B(t) = (1-t)^2 P_0 + 2(1-t)tP_1 + t^2 P_2$, $t \in [0, 1]$
- Cubic Bézier curve: $B(t) = (1-t)^3 P_0 + 3(1-t)^2 tP_1 + 3(1-t)t^2 P_2 + t^3 P_3$, $t \in [0, 1]$

where P_is are the control points.

After acquiring an ideal trajectory from the shortest path on the roadmap, approximate trajectory must be generated by using a trajectory approximation method and Bézier curve fitting algorithms. We used Equal Error Heuristic (EEH) [20] as the approximation method for the framework and M. Khan's Bézier curve fitting algorithms in [22] and [23]. For EEH, we set the application's accuracy constraint as 500 pixels, i.e., the approximate trajectory cannot be more than 500 pixels away from the ideal trajectory.

C. Packet Forwarding

When forwarding a packet along a trajectory curve, several different ways are possible for selecting the next neighbor: the neighbor closest to the destination (i.e., farther along the trajectory) or the neighbor that yields the least amount of deviation from the curve. Depending on the application-specific constraint, a particular forwarding technique could be adopted. For instance, if the application is sensitive to delay, choosing the neighbor closest to the destination would be better. Since we are using path accuracy as the application's desired constraint, we adopted the Closest to Curve (CTC) method [4] to pick the next hop. CTC works as follows: Assume node A that decodes the trajectory curve from a packet she needs to forward. First, A calculates its position on the curve by finding the t_A value of the closest point on the curve. Then it calculates t_i values for each neighbor N_i on the curve. Note that t_is will all be in $[0, 1]$, with greater meaning farther along the curve. Then we pick the neighbor which is closest in distance to the curve where $t_i \in [0, 1]$ and $t_i > t_A$. A also checks if any of its neighbors are on the next segment of the trajectory if exists. In this case it picks the closest neighbor to the next trajectory segment.

D. Merging Congestion Indications to Roadmaps

Initially, transmission attempts by the source node are more likely to fail because roadmap edges are set to initial values and do not accurately reflect the network state. However, the roadmap will be shaped by a feedback mechanism and become a more accurate and balanced representation of the network. Mobility, changes in the network traffic and other dynamics of the environment affect the robustness of this balanced state of the roadmap. There are two types of feedback from the network to the source nodes:

Feedback 1 – Hitting a Void Area: The first type of feedback comes from nodes which cannot find a feasible node on the trajectory to forward a packet to. Such areas where there are no nodes are called *void areas*. In that case, the node drops the packet, inverts the trajectory of the dropped packet and sends a feedback packet, that has a high priority, back to the source node. Feedback packets include the fail location, so, the source node will stop the flows traversing that fail location. Also, the source node marks the roadmap edges close to the fail location by increasing their values. Finally, the source node calculates a new trajectory and resumes transmission over it.

Feedback 2 – Hitting a Congested Area: Second type of feedback comes from congested nodes. When a node drops a packet due to congestion, it broadcasts a feedback packet

including the information of congested location. Since such broadcasts are expensive, the nodes ignore packet drops for a time period after broadcasting. Upon receiving these packets, the source nodes mark the roadmap edges around the congested location. Then, for every flow using those edges, they generate a new trajectory with a 50% probability.

The feedback mechanism tells the source nodes whether a packet drop is caused by congestion or void area. However, a source node cannot tell whether the condition is permanent or temporary. The congestion may be resolved after some of the source nodes change their trajectories or new nodes may arrive at the void areas in the future. Therefore, all cases are assumed as temporary. Over time, the source nodes gradually decrease all roadmap edges' values.

TABLE I. NETWORK PROTOCOL PARAMETERS

Physical Layer	
Chanel bit rate	$1Mbps$
Propagation delay	$1\mu s$
Physical header	128 bits
MAC Layer	
Slot time	$50\mu s$
SIFS	$28\mu s$
DIFS	$128\mu s$
CTS timeout	$300\mu s$
Minimum backoff window	$16\mu s$
Maximum backoff window	$1024\mu s$
Maximum retries before packet is dropped	10
ACK packet length	Physical header $+112$ bits
RTS packet length	Physical header $+160$ bits
CTS packet length	Physical header $+112$ bits
MAC header	272 bits
Network Layer	
Packet payload	8184 bits
Feedback packet payload	128 bits

IV. SIMULATION SETUP

To gain insights into how well our roadmap-based approach to path selection performs, we ran simulation experiments. Among the metrics of particular interest are the evenness of *load balancing* across network nodes, *aggregate throughput*, and average *packet delivery time* (i.e., end-to-end delay). We compared our roadmap-based TBR against GPSR [24], which is a shortest-path routing protocol and provides an excellent benchmark for comparative evaluation.

We implemented IEEE 802.11 MAC and PHY specifications with CSMA/CA [25] and integrated TBR and GPSR as network layer protocols. Finally, we implemented UDP [26] as the transport layer protocol. Table I shows the parameters we used in different layers of the network protocols.

TABLE II. ROADMAP VALUES

Minimum edge length (d_c)	120
Maximum edge length (d_f)	240
Initial edge value (E_0)	0.001
Each packet increment (E_+)	0.01
Decrement over time (E_-)	0.01
Decrement period	$800\mu s$
Void area and congestion penalty (E_p)	1

We used the values in Table II for roadmap implementation. We set d_c to 120 and d_f to 240. Each edge's initial weight (E_0) is set to 0.001. Whenever a packet is sent over a trajectory, the weight of each edge along the trajectory is increased by E_+=0.01. When a feedback packet is received, either caused by a congestion or a void area, all edges around the bad location gain E_p=1 to their weights. At every $800\mu s$, all edges weighing at least E_0 get their weights reduced by E_-=0.01.

TABLE III. CONSTANT AND VARIABLE SIMULATION PARAMETERS

Field height	1500 pixels
Field width	1500 pixels
Node transmission range	150 pixels
Simulation reruns	16
Trajectory life	50
Network density (avg # of neighbors)	$10, 15, 20, 25$
Number of nodes (based on network density)	$114, 171, 229, 286$
Number of traffic flows	$3, 5, 10$
Packet buffer size at each node	$5, 10, \ldots, 50$
Simulation run time	$20s$
Packet traffic rate	$160Kbps$
Packet generation period	$51150\mu s$
Packet generation frequency	$\sim$20 packets per second

Table III shows the other parameters of the simulation. Nodes are identical with a transmission range of 150 pixels and randomly distributed on a planar field of size 1500×1500 pixels. Trajectory life is set to 50, which means that the source node will change its trajectory after sending 50 packets over it. We assigned three variable parameters to each simulation run: network density, number of traffic flows, and packet buffer size. We defined network density by the average number of neighbors of a node. We used $10, 15, 20$ and 25 as the network density. We initiated 3, 5 and 10 traffic flows to experiment with different traffic load. Packet buffer size is the number of packets that a node can store before it transmits them. We assumed drop-tail queuing behavior.

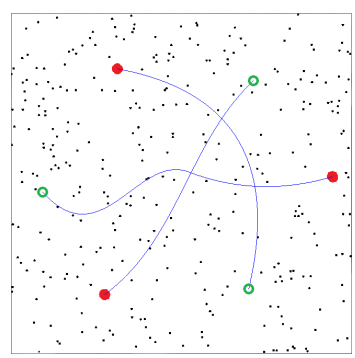


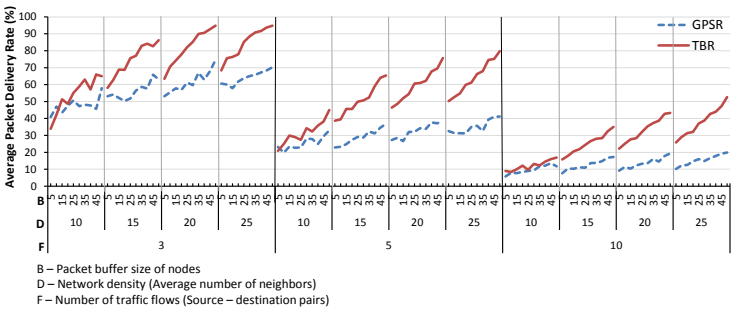
Fig. 3. Source (green circle) and destination (red disk) nodes are positioned evenly on a circle. A traffic is generated between a source node and a destination node located on the opposite side of the circle.

We placed source and destination nodes on positions landing on a circle, as shown in Figure 3, so that the flows will collide. To do so, we drew an imaginary circle with diameter of 1200 pixels, right in the middle of the field. For n flows, we pick $2n$ locations evenly distributed on the circle. Each location on this side of the circle is paired with the location on the opposite side.

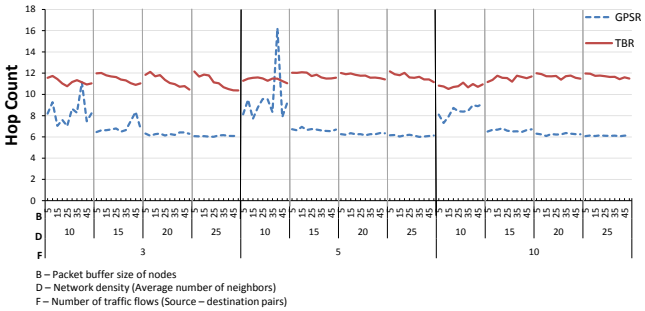
We ran each simulation for $20s$ with traffic rate at $160Kbps$ for each flow. This translates to $\sim$20 packets per second per traffic flow. For each combination of parameters, we made 16 reruns with different seeds and used the average in plots.

V. RESULTS

We extracted throughput, packet delivery time, hop count and load balancing results from simulations for TBR and GPSR, and compared their performance by these metrics.



(a) Average throughput



(b) Average hop count

Fig. 4. TBR delivers higher throughput in most scenarios while it usually uses more hops than GPSR.

1) Throughput: Figure 4(a) shows successful packet delivery rates. TBR outperforms GPSR in almost every scenario. As the number of traffic flows increase, congestion also increases. High congestion greatly reduces performance of both protocols. When TBR hits a void area, packets on the bad trajectory are dropped. On the other hand, GPSR starts with the greedy forwarding mode for each packet. Because of this difference, GPSR sometimes perform slightly better than TBR on sparse networks.

2) Hop Count: Figure 4(b) shows the average number of hops that the packets take until reaching their destinations. Trajectories become longer when they go around the central area which increases the hop count. Increasing the buffer size reduces the congestion, samely hop counts.The spikes are caused by void areas which triggers the perimeter mode in GPSR.

3) Packet Delivery Time: Figure 5(a) shows average packet delivery times. As the number of flows increases, more packets are sent into the network. Therefore, nodes have to store the packets in their buffers more often which results in packet delays. Another reason for huge delays are large buffer sizes. When a buffer is large in size, the packets have to wait longer for their turn.

4) Load Balancing: Figure 5(b) shows how well TBR and GPSR can distribute the work load around the network. We define work load as *the total number of packets received and transmitted*. We count both the reception and the transmission of packets towards the work load since transmissions do contribute to the wireless channel usage. We specifically look at the standard deviation of the work load across the nodes of the network. A lower standard deviation of the work load means that the load is balanced better across the nodes.

Figure 6 shows the heat map of the nodes in a simulation run where network density is 20, number of flows are 10 and buffer size is 25. The results consistently show higher standard deviation for GPSR. As the number of traffic flows increase, the work load accumulates on a small number of nodes at the intersection point. When the buffer is full, a packet is dropped. Therefore, some source nodes change their trajectories and use different nodes on the next. This is more clear in Figure 6 that TBR better distributes the load over the whole network. The number of nodes involved in the traffic is much higher compared to GPSR. Also dark disks in GPSR show us that very few nodes take on the majority of the work load on themselves.

Results from our simulation experiments clearly show that the roadmap-based TBR outperforms GPSR in terms of aggregate throughput Figure 4(a) by better balancing the traffic load on the network. This confirms our intuition that path/trajectory selection over an abstract roadmap is capable of steering the traffic flows so that the network traffic is engineered for higher throughput. Of course, this gain in throughput means larger E2E delays in the traffic as shown in Figure 5(a). However, the good news is that the extra delay is not significant while the gains in throughput are large.

VI. Summary and Future Work

We presented a roadmap-based framework to perform E2E TE by using TBR. We detailed how to generate ideal trajectories avoiding obstacles and void areas while trying to balance the traffic load in a multi-hop wireless network. We compared our roadmap-based framework to GPSR – the most widely used benchmark protocol for shortest-path routing in multi-hop wireless networks. Our approach showed significantly higher throughput in almost every condition. Workload heatmap showed that roadmap-based TBR distributes the load over the network, while GPSR keeps the load on a small number of nodes at the center of the topology. On the other hand, TBR packets take slightly longer time to reach their destinations because they take longer paths compared to GPSR packets. This tradeoff was expected and it proved to be good for our framework; because, the throughput gains of TBR were very high when the amount of additional delay was considered. For example, TBR attained 2.75 times more throughput for cases where its packets had to go through 25% additional delay.

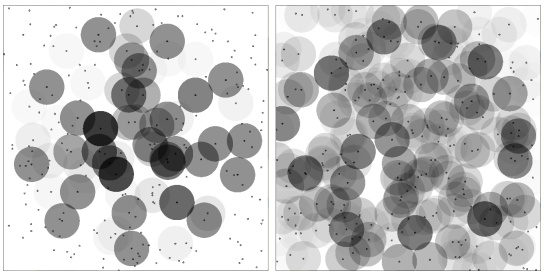
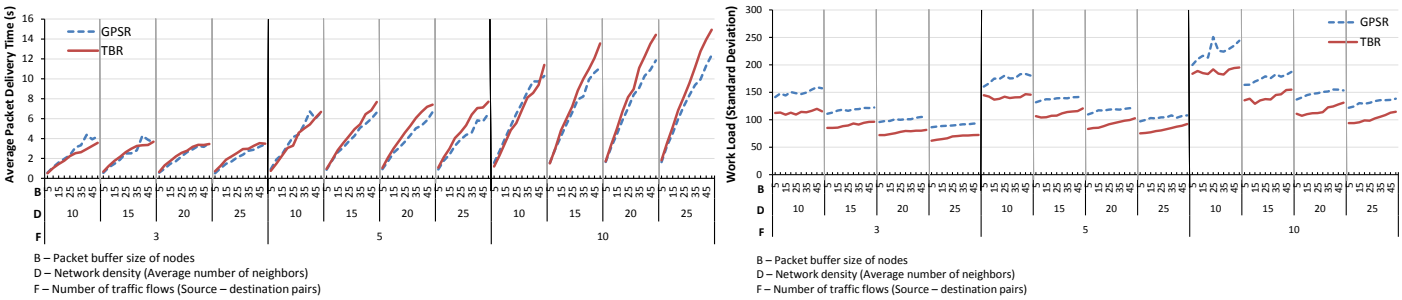


Fig. 6. Load balancing for GPSR (left) and TBR (right): Each circle represents a node. Opaqueness of circles shows the workload (# of pkts received + transmitted). TBR distributes the load better than GPSR. ($D = 20, F = 10, B = 25$)

Our trajectory planning techniques can be improved in several ways. Instead of distributing roadmap vertices randomly on the field, a common pattern can be used. A roadmap



(a) Average packet delivery time

(b) Std deviation of avg workload (# of pkts received + transmitted)

Fig. 5. TBR packets take more time to reach their destinations because of longer trajectories. TBR distributes load over the network while GPSR does not.

formed of triangles, squares or hexagons may yield higher performance. The edges are formed according to fixed d_c and d_f values. We also fixed other parameters such as E_0, E_+ and E_-. Again, different values and varying parameters may give better performance. These parameters can also be functions of other parameters such as the network density and node transmission range. Also, improvements to the feedback mechanism can help a good deal for preventing congestion. For instance, probing the congestion status along trajectory before sending data packets would be a more proactive way of collecting network state and will generate better E2E throughput.

The current implementation generated one roadmap for each node, and the nodes do not share their roadmaps with each other. A better approach would be to generate a global roadmap and make the nodes share their updates with each other. Last, but not the least, stability and convergence properties of E2E TE using roadmap-based TBR needs to be studied.

Acknowledgement

This work was supported in part by NSF award 1321069.

References

[1] L. Tassiulas, "Adaptive back-pressure congestion control based on local information," *Automatic Control, IEEE Transactions on*, vol. 40, no. 2, pp. 236–250, Feb 1995.

[2] E.Rosen and et al., "Multiprotocol label switching architecture," RFC 3031, January 2001.

[3] D. Niculescu and B. Nath, "Trajectory based forwarding and its applications," in *Proceedings of ACM MOBICOM*, 2003.

[4] M. Yuksel, R. Pradhan, and S. Kalyanaraman, "An implementation framework for trajectory-based forwarding in ad-hoc networks," *Ad Hoc Networks, Elsevier Science*, vol. 4, no. 1, pp. 125–137, January 2006.

[5] L. E. Kavraki, P. Svestka, L. E. K. P. Vestka, J. claude Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 566–580, 1996.

[6] S. Bhattacharya, N. Atay, G. Alankus, C. Lu, O. B. Bayazit, and G. catalin Roman, "Roadmap query for sensor network assisted navigation in dynamic environments," in *in International Conference on Distributed Computing in Sensor Systems (DCOSS06*, 2006.

[7] G. Alankus, N. Atay, C. Lu, and O. Bayazit, "Adaptive embedded roadmaps for sensor networks," in *Robotics and Automation, 2007 IEEE International Conference on*, April 2007, pp. 3645–3652.

[8] B. P. et al., *Global Positioning System: Theory and Application*. Progress in Astronautics and Aeronautics, 1996, vol. 163.

[9] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu, "OpenDHT: A public dht service and its users," in *Proceedigs of ACM SIGCOMM 2005*, 2005.

[10] C. Petrioli, M. Nati, P. Casari, M. Zorzi, and S. Basagni, "Alba-r: Load-balancing geographic routing around connectivity holes in wireless sensor networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 25, no. 3, pp. 529–539, March 2014.

[11] L. Ying, S. Shakkottai, A. Reddy, and S. Liu, "On combining shortest-path and back-pressure routing over multihop wireless networks," *IEEE/ACM Trans. Netw.*, vol. 19, no. 3, pp. 841–854, Jun. 2011.

[12] L. Ying, R. Srikant, D. Towsley, and S. Liu, "Cluster-based back-pressure routing algorithm," *Networking, IEEE/ACM Transactions on*, vol. 19, no. 6, pp. 1773–1786, Dec 2011.

[13] S. Jung, M. Kserawi, D. Lee, and J.-K. K. Rhee, "Distributed potential field based routing and autonomous load balancing for wireless mesh networks," *Comm. Letters.*, vol. 13, no. 6, pp. 429–431, Jun. 2009.

[14] A. Mei, N. Piroso, and B. Vavala, "Fine grained load balancing in multi-hop wireless networks," *Journal of Parallel and Distributed Computing*, vol. 72, no. 4, pp. 475–488, 2012.

[15] E. Jones, M. Karsten, and P. Ward, "Multipath load balancing in multi-hop wireless networks," in *Wireless And Mobile Computing, Networking And Communications, 2005. (WiMob'2005), IEEE International Conference on*, vol. 2, Aug 2005, pp. 158–166 Vol. 2.

[16] B. Scheuermann, M. Transier, C. Lochert, M. Mauve, and W. Effelsberg, "Backpressure multicast congestion control in mobile ad-hoc networks," in *Proceedings of the 2007 ACM CoNEXT Conference*, ser. CoNEXT '07. New York, NY, USA: ACM, 2007, pp. 23:1–23:11.

[17] T. He, J. Stankovic, C. Lu, and T. Abdelzaher, "Speed: a stateless protocol for real-time communication in sensor networks," in *Distributed Computing Systems, 2003. Proceedings. 23rd International Conference on*, May 2003, pp. 46–55.

[18] N. Pantazis, S. Nikolidakis, and D. Vergados, "Energy-efficient routing protocols in wireless sensor networks: A survey," *Communications Surveys Tutorials, IEEE*, vol. 15, no. 2, pp. 551–591, Second 2013.

[19] M. O. Kilavuz and M. Yuksel, "Minimizing multi-hop wireless routing state under application-based accuracy constraints," in *Proceedings of IEEE MASS*, Atlanta, GA, September 2008.

[20] M.O.Kilavuz and M.Yuksel, "Path approximation for multi-hop wireless routing under application-based accuracy constraints," *Computer Networks*, vol. 56, no. 1, pp. 345–364, January 2012.

[21] P. J. Schneider, "Graphics gems," A. S. Glassner, Ed. San Diego, CA, USA: Academic Press Professional, Inc., 1990, ch. Solving the Nearest-point-on-curve Problem, pp. 607–611.

[22] M. Khan, "A new method for video data compression by quadratic bezier curve fitting," *Signal, Image and Video Processing*, vol. 6, no. 1, pp. 19–24, 2012.

[23] M.Khan, "Approximation of data using cubic bezier curve least square fitting," 2007, http://www.mathworks.com/matlabcentral/fileexchange/15542-cubic-bezier-least-square-fitting.

[24] B. Karp and H. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in *Proceedings of ACM/IEEE MobiCom 2000*, August 2000.

[25] "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," IEEE Standard 802.11, June 1999.

[26] J. Postel, "User datagram protocol," Internet Engineering Task Force, RFC 768, August 1980.