Trajectory Based Forwarding and Its Applications

Dragoş Niculescu and Badri Nath
Rutgers University
DATAMAN Lab
{dnicules,badri}@cs.rutgers.edu

ABSTRACT

Trajectory based forwarding (TBF) is a novel method to forward packets in a dense ad hoc network that makes it possible to route a packet along a predefined curve. It is a hybrid between source based routing and Cartesian forwarding in that the trajectory is set by the source, but the forwarding decision is based on the relationship to the trajectory rather than names of intermediate nodes. The fundamental aspects of TBF are: it decouples path naming from the actual path; it provides cheap path diversity; it trades off communication for computation. These aspects address the double scalability issue with respect to mobility rate and network size. In addition, TBF provides a common framework for many services such as: broadcasting, discovery, unicast, multicast and multipath routing in ad hoc networks. TBF requires that nodes know their position relative to a coordinate system. While a global coordinate system afforded by a system such as GPS would be ideal, approximate positioning methods provided by other algorithms are also usable.

Categories and Subject Descriptors

C.2.1 [**Network architecture and design**]: Wireless communication; C.2.2 [**Network protocols**]: Routing protocols

Keywords

ad hoc networks, trajectory based forwarding, routing, multipath, broadcasting, positioning

1. INTRODUCTION

Recent advances in wireless communication devices, sensors, hardware (MEMS) technology make it possible to envision large scale dense ad-hoc network acting as high resolution eyes and ears of the surrounding physical space. Examples of such vision include smartdust [1], dataspaces [2, 3], sensitive skin [4], or disposable networks where it is possible that many of the nodes of the network can be sprayed, dropped, mixed in the material or embedded in the infrastructure.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
MobiCom'03, September 14–19, 2003, San Diego, California, USA.
Copyright 2003 ACM 1-58113-753-2/03/0009 ...$5.00.

These networks are characterized by a large number of energy-constrained, unattended nodes. Beside the algorithmic aspects dictated by their sheer scale, the emphasis on energy efficient algorithms require that nodes often go into doze mode or sleep mode resulting in a very dynamic network topology. These characteristics require us to rethink the way many of the networking functions can be implemented [5]. One of the fundamental networking functions is routing. Routing always has been treated as sending packets along route paths described by a discrete set of points. Routing algorithms used in the fixed networks and those proposed by the MANET community for ad-hoc networks are aimed at resource-rich, relatively stable networks.

In this paper, we propose a new forwarding paradigm, Trajectory based forwarding (TBF) which addresses the issue of scalability and dynamic network topology. This is fundamentally new approach to routing in "dense matter" where the route path is specified and treated as a continuous function as opposed to a discrete set of points. The transition from **a discrete view of route paths to a continuous view of route paths** is only natural as we move from dealing with sparse networks to dealing with dense networks. The key idea in the approach is to embed a trajectory in the packet and then let the intermediate nodes forward packets to those nodes that lie more or less on the trajectory. Representing route paths as trajectories is an efficient scalable encoding technique for dense networks. Since a trajectory does not explicitly encode the nodes in the path, it is to a large extent impervious to changes in specific nodes that make up the topology. We believe that trajectories are a natural namespace to describe route paths when the topology of the network matches the topography of the physical surroundings in which it is deployed which by very definition is embedded computing. Here, the physical paths traversed by packets mirror the underlying shape of the physical space that is being queried. Further, forwarding packets along trajectories can be very effective in implementing many networking functions when standard bootstrapping or configuration services are not available, as will be the case in disposable networks where nodes are thrown or dropped to form a one-time use network.

Although Cartesian routing [6] offers the possibility of routing packets based on positions, it does so only on straight lines between source and destination. There are many practical network services that require routing along routes possibly other than the shortest path. One such example is multipath routing, which may be employed by a source to increase bandwidth, or resilience of communication. Rout-

ing along the shortest path is not always the best option in wired networks [7]. In sensor networks the same problem manifests itself as potential network partitioning due to battery overuse along popular shortest paths. Communication over alternate paths must be therefore used as a load balancing method in order to achieve more uniform battery depletion. Finally, non straight trajectories are necessary to describe unicast routes in a network where straight line forwarding is not possible due to obstacles, holes in connectivity, or other criteria, such as security requirements.

TBF has a number of features that make it an ideal candidate for a low level primitive in any ad hoc network.

1. it decouples the the path name from the path itself. This is the most critical aspect in a dense network, where intermediate nodes between source and destination might move, go into doze mode or fail, thereby rendering a discrete source based path useless.

2. the specification of the trajectory is independent of the name of the destination. This makes TBF usable as a routing support, when the destination is indicated, as a discovery support primitive, when the destination is not known, or as a flooding replacement.

3. it provides cheap path diversity, when compared to flooding based traditional methods of finding alternate paths.

4. it trades off communication for computation, by declaring paths instead of searching them. This is a desirable tradeoff, considering the four orders of magnitude difference between the cost of sending a wireless packet and executing an instruction [1].

5. it may be assisted by various functionalities available in the nodes. Ideally, each node would be equipped with a GPS receiver, case in which nodes closest to the indicated trajectory will forward the packet. If, however, GPS is not available (such as non line of sight scenarios, or lack of sufficient precision) TBF may use approximate positions given by positioning algorithms [8, 9, 10, 11] that are based on nodes' other abilities to sense their neighbors (ranging, angle of arrival, compass).

Besides simple unicast, trajectory routing and forwarding have significant advantages for many other important network functions such as broadcasting, discovery, multipath, multicast and broadcast, path resilience. In this paper, we focus on issues related to trajectory forwarding in networks with and without the availability of node positions, and identify a number of research challenges related to trajectories in ad hoc networks.

The rest of the paper is organized as follows: the next section reviews related work, section 3 details our proposed approach, the network model assumptions, various forwarding methods and implementation issues; section 4 reviews applications that benefit from an implementation under the TBF framework; section 5 debates problems TBF faces under adverse conditions such as reduced density and lack of positioning capabilities. Section 6 summarizes the current status of the project and mentions some challenging issues and possible future work, and we summarize with some concluding remarks in section 7.

2. RELATED WORK

There have been significant efforts to improve routing in both fixed and mobile networks when position is available. Such methods, in which node spatial positions are essential to the method, are branded "position centric". Geographic routing [12] is a hierarchical scheme where each router is responsible for a polygonal region possibly subpartitioned into disjoint polygons assigned to other routers. This routing scheme provides an infrastructure that can be embedded in IP, and can deliver messages to specific geographic regions. Cartesian routing [6] is a greedy method that chooses a next hop that provides most progress towards the destination. It is a particular case of TBF, and both are representative for position centric routing. LAR (Location Aided Routing) [13] is another (position centric) scheme that implements restricted area flooding in order to reduce the cost of discovery when the uncertainty about a destination is limited. It uses a phase of source based routing and a phase of controlled flooding.

Cartesian routing, and other greedy methods derived from it do not guarantee the delivery of the packets. The problem is usually addressed by planarizing the network graph and applying detour algorithms, such as FACE [14], GPSR [15], or GOAFR+ [16], that avoid obstacles using the "right hand rule" strategy that work well for straight line delivery.

The other big category, inherited from wired networks, is "node centric" - destinations and intermediate forwarding entities are names of nodes. DSR (Dynamic Source Routing) [17] is a form of source based routing used in MANET [18], featuring a route discovery phase based on flooding, and routes completely specified in packet headers. Terminode routing [19] is also a source based method, but uses anchors instead of intermediate nodes. It is close in spirit to our proposed method, but is discrete in the representation of the path. The method may still entail large overheads for long paths that might otherwise have a compact parametric representation.

In order to use position centric approaches, node positions are necessary, but a locations service is also necessary to translate node addresses into coordinates. GLS (Grid Location Service) [20] implements a naming service that allows node centric applications to run on top of geographic and cartesian routing. A source can find the coordinates of the destination node from the location service and then use geographic or cartesian routing to route to that destination. Other location services include DREAM [21], which updates locations with remote communication pairs based on angular drift, and [22], which makes use of Bloom filters to decide if a mobile is in a certain area.

A more recent approach is "data centric", pioneered in [5], in which routing is driven by *interests,* describing the meaning of data transfered. In a sensor network, multipath routing may be useful in providing resilience [23].

TBF can be used to enhance or complement all the node centric, position centric and data centric mechanisms, or to replace expensive energy-wise parts of them, such as flooding based discovery.

3. TBF DESCRIPTION

TBF is a hybrid technique combining source based routing [17] and Cartesian forwarding [6], but uses a continuous representation of the route. Like in source based routing,

Figure 1: TBF layer

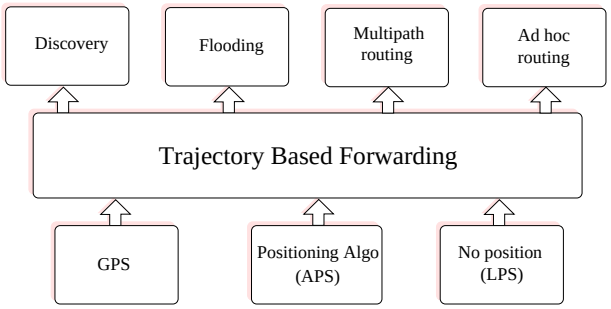
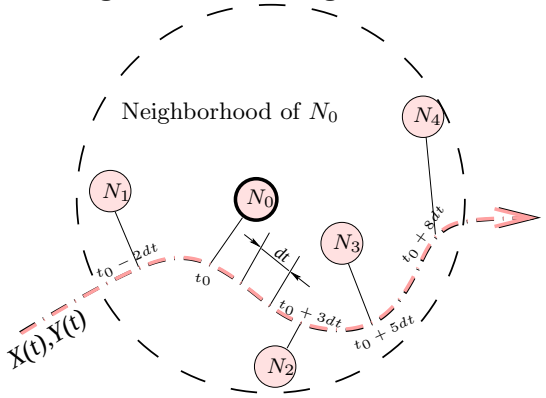


Figure 2: Forwarding on a curve



the path is indicated by the source, but without actually specifying all the intermediate nodes. Like in Cartesian forwarding, decisions taken at each node are greedy, but are not based on distance to destination - the measure is the distance to the desired trajectory. Source based routing has the advantage that intermediate nodes are relieved of using and maintaining large forwarding tables, but it has the disadvantage of the packet overhead increasing with the path length. Cartesian routing uses positions to get rid of the routing tables, but defines one single forwarding policy: greedy, along a straight line.

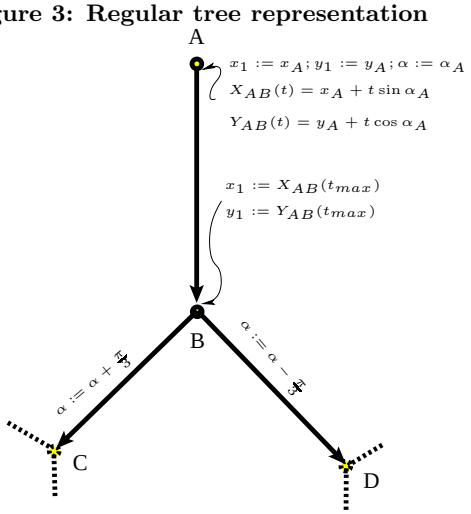
TBF gets the best of the two methods: packets follow a trajectory established at the source, but each forwarding node takes a greedy decision to infer the next hop based on local position information, while the overhead of representing the trajectory does not depend on path length. In a network where node positions are known, the packet may be forwarded to the neighbor that is geographically closest to the desired trajectory indicated by the source node. If the destination node is known, the trajectory followed by the packet might be a line, and the method reduces to cartesian forwarding. In the general case, however, we envision a larger array of applications including ad hoc routing, discovery, flooding, and multipath routing, as shown in figure 1. Trajectory based forwarding (TBF) requires that nodes be positioned relative to a global coordinate system or a relative coordinate system. The strength of TBF lies in the flexibility of being able to work over a wide variety of positioning systems. In fact, TBF can be seen as a middle layer between global [24], ad hoc [8, 25, 9, 10] and local [11] position providing services, and many network management services.

3.1 Forwarding methods

In the TBF framework, routing a packet requires that a trajectory be specified along which a packet can be forwarded. The trajectory is usually decided by the source and we will assume that it is expressed in parametric form $X(t), Y(t)$. For example, to route along a line with slope α passing through the source with coordinates x_1, y_1, the trajectory would be described by $X(t) = x_1 + t\cos(\alpha)$; $Y(t) = y_1 + t\sin(\alpha)$. α, x_1, y_1 are constants, and the parameter t actually describes euclidean distance traveled along the line. It is convenient, for the simplicity of the explanation to assume that t indicates the distance on the curve, but it need not be so in the general case. The neighborhood of a node N_0 (figure 2) is defined as the portion of the curve and the nodes that are within a certain distance form N_0, shown by a dashed line in the figure. In the simplest case, the neighborhood could be the smallest circle enclosing all N_0's one hop neighbors. In a network in which node positions are known, the main question is how to choose a next hop that best approximates the trajectory. Assume node N_0 receives a packet with the trajectory indicated by the curve $X(t), Y(t)$ and the value t_0 that corresponds to the point on the curve that is closest to N_0. Using sampling of the curve at dt spaced intervals, indicated by dots in the dashed trajectory curve, N_0 can compute all the points of the curve that reside inside its neighborhood. For all neighbors $N_1..N_4$, their corresponding closest points on the curve are t_0, t_0+3dt, t_0+5dt, t_0+8dt. When referring to curve fitting, these values are called residuals. In fact, the mentioned method computes an estimation of the residuals, instead of the true ones, which would require either infinite resolution ($dt \to 0$), or usage of higher derivatives of $X(t)$ and $Y(t)$. Since choosing a next hop for the packet should be towards advancement on the trajectory, only the portion of the curve with $t > t_0$ is considered. For this reason, node N_1 receives t_0 as the closest point, instead of $t_0 - 2dt$, which would be closer to the perpendicular from N_1 onto the curve. Several policies of choosing a next hop are possible:

- "minimum deviation": choose the node closest to the curve, with the minimum residual. This policy would favor node N_2 and would tend to produce a lower deviation from the ideal trajectory;
- most forwarding within radius (MFR) [26], choosing N_4. This policy should also be controlled by a threshold of a maximum acceptable residual, in order to limit the drifting of the achieved trajectory. It would produce paths with fewer hops than the previous policy, but with higher deviation from the ideal trajectory;
- centroid of the feasible set, favoring N_3: the centroid is a way to uniformly designate clusters along the trajectory, and a quick way to determine the state of highly dense networks;
- the node with most battery left;
- randomly choose between best three: useful when node positions are imperfect, or when it may be necessary to route around obstacles;

Figure 3: Regular tree representation



- in mobile networks a forwarding policy that might provide better results would be to choose the next hop which promises to advance along the trajectory, or one that is expected to have the least mobility in the future.

3.2 Trajectory specification/encoding

There are a number of choices in representing a trajectory: functional, equational, or parametric representation. Functional representation (e.g. $Y = f(X)$) cannot be used to specify all types of curves (for example vertical lines). Equational representation (e.g. $X^2 + Y^2 = R^2$) requires explicit solution to determine the points on the curve. Parametric representation (e.g. $X = X(t), Y = Y(t)$) is ideally suited for the purpose of forwarding. The parameter t of the curve is a natural metric to measure the forward progress along the path and can be linked to either length traveled on the curve, or hop count.

The next issue is how to encode a trajectory that can have several parameters which have to be interpreted by nodes. One approach is to have a tabular representation where the nodes know how to interpret the fields given a well know set of trajectories. The line mentioned in subsection 3.1 would be represented by a tuple $(line, x_1, y_1, \alpha)$, possibly with additional bits describing the forwarding policy. The first item describes the type of trajectory, followed by the specific parameters. A node would have a fixed dictionary of available trajectories – this method provides the lowest packet overhead, but limited flexibility.

Complex trajectories can have multiple components or a given trajectory can be specified as a number of simple component such as Fourier components. The more Fourier components are specified in the packet, the better the accuracy of the trajectory is. There is an interesting tradeoff between the accuracy of the curve and the overhead of specifying the components and interpreting them. Other possibilities of encoding of the parametric curve include compiled form (ready to be executed, as in active networking), or reverse polish notation (ready to be interpreted). In our current implementation on Mica motes, we used the latter, for the increased flexibility.

For multicast, a source based method like TBF should store the entire distribution tree in each packet, which could significantly increase overhead. However, in certain cases this overhead may be reduced. When the tree has a regular shape, like a self repeating structure, its recursive representation is actually very compact. In figure 3, line AB is defined in its parametric form, but parameterizing what we previously used as constants: $x_1 := x_A$; $y_1 := y_A$ and $\alpha := \alpha_A$, with an additional parameter t_{max} to indicate the packet duplication point B. Therefore, the description of segment AB is:

$$X_{AB}(t) = x_A + t\cos(\alpha_A)$$
$$Y_{AB}(t) = y_A + t\sin(\alpha_A)$$

When $t \in (0, t_{max})$, forwarding works using normal unicast in the interval AB. A node that is close to B, with the coordinates $B(X_{AB}(t_{max}), Y_{AB}(t_{max}))$, performs the duplication task required in multicast, and prepares two new trajectories. These are BC and BD, in fact lines of length t_{max}, just like their parent, but with different parameters. t starts fresh from 0, for both these two segments, and stays in the same interval $t \in (0, t_{max})$, but with new equations:

$$X_{BC}(t) = x_B + t\cos(\alpha_A + \tfrac{\pi}{3})$$
$$Y_{BC}(t) = y_B + t\sin(\alpha_A + \tfrac{\pi}{3})$$

$$X_{BD}(t) = x_B + t\cos(\alpha_A - \tfrac{\pi}{3})$$
$$Y_{BD}(t) = y_B + t\sin(\alpha_A - \tfrac{\pi}{3})$$

Their starting point is B, the ending point of the parent, with coordinates $x_B = X_{AB}(t_{max})$, $y_B = Y_{AB}(t_{max})$, and their angles diverge in this example by $\frac{2\pi}{3}$ from each other, being $\frac{\pi}{3}$ sideways of their parent. The node responsible for duplication at point B receives a TBF packet with the tuple $(line, x_A, y_A, \alpha_A, t_{max})$, which describes $X_{AB}(t)$ and $Y_{AB}(t)$, and creates two new TBF packets described by tuples:

- $(line, X_{AB}(t_{max}), Y_{AB}(t_{max}), \alpha_A + \frac{\pi}{3}, t_{max})$
- $(line, X_{AB}(t_{max}), Y_{AB}(t_{max}), \alpha_A - \frac{\pi}{3}, t_{max})$

The packet splits behave like recursive calls in the compact representation of this fractal structure, meaning that the splitting process will continue at points C and D. The representation is compact in that for describing an arbitrarily large tree of this shape, just the three tuples mentioned need be transmitted as trajectory encoding. The structure produced by this particular example is in fact a honeycomb structure, shown in figure 8b. The recursive definition of the structure will eventually overlap itself, therefore marking bits are necessary in order to stop the forwarding. This is also discussed in the context of stateful versus stateless broadcasting (section 4.4).

Even in the case of an arbitrary tree the overhead may be reduced by a simple pruning scheme. This is the case when the tree must be described in its entirety at the source, for example by specifying the splitting points. In figure 4, the tree describing the leafs (multicast receivers) and the splitting points gets stripped as the forwarding proceeds downstream. The amount of total overhead as a function of n, the total number of receivers, is therefore $O(n \log n)$ in the average case (for balanced trees), as opposed to $O(n^2)$ when sending the full tree on each branch.

Figure 4: Arbitrary tree pruning

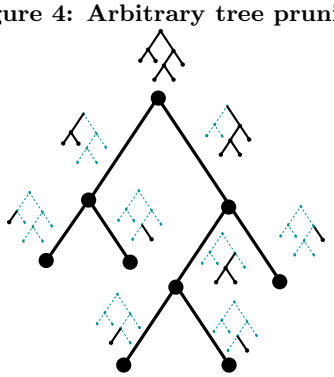


Figure 5: Examples of trajectory routing and forwarding

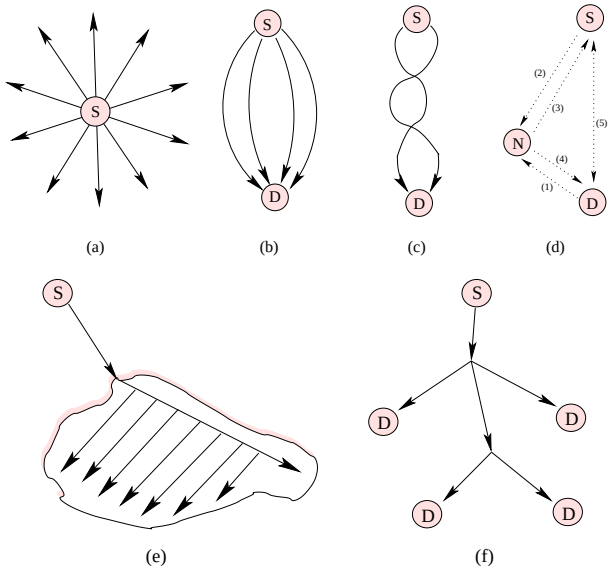
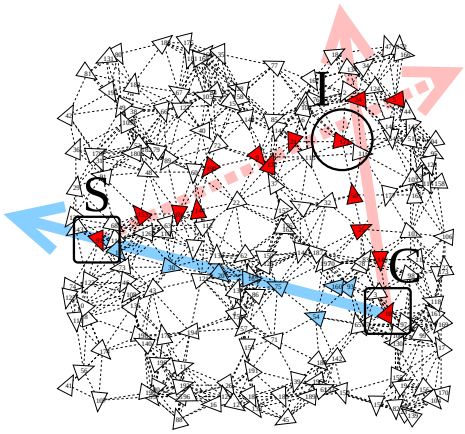


Figure 6: Discovery example



4. APPLICATIONS OF TBF

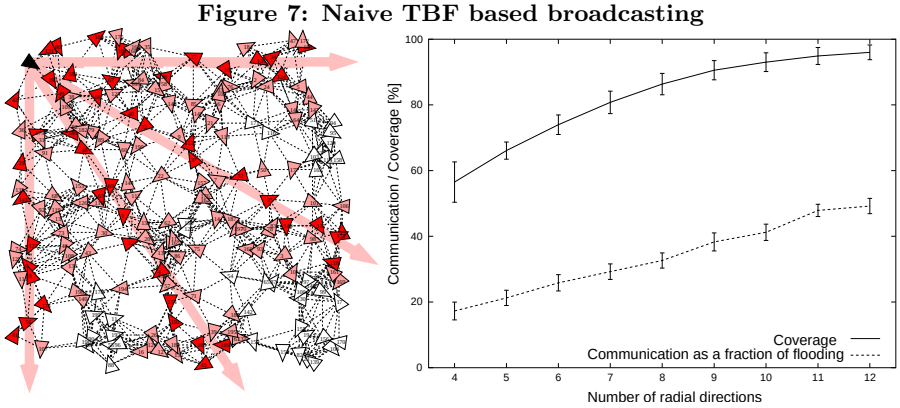
Having discussed the implementation details of TBF, we shall now investigate some of the applications that would benefit from an implementation under the TBF framework. There is a wide variety of trajectory shapes that can be used in applications, but a broad classification of trajectories may be into simple or composed. Simple trajectories describe a single continuous curve, and, in the context of routing, are used for unicast. Composed trajectories describe several, spatially different curves. They may also be used for unicast in an anchor based fashion, when a complicated trajectory is described as a list of simpler trajectories. Composed trajectories have a more obvious use in broadcast and multicast, where a unique curve is less appropriate.

A sampling of TBF based applications is shown in figure 5. A naive broadcasting scheme based on trajectories uses a number of radial outgoing lines that are reasonably close to each other to achieve a similar effect without all the communication overhead involved by receiving duplicates in classical flooding (figure 5a). More generally, a source would indicate the directions and the lengths of the lines that would achieve a satisfactory coverage (figure 5e). Coverage relies on the typical broadcast property of the wireless medium, in which several nodes overhear the packet being forwarded. Recovery from failure often involves multipath routing from a source to a destination. In a sensor network, both disjoint (figure 5b) and braided (figure 5c) paths are useful in providing resilience [23]. A simple five step discovery scheme (figure 5d) based on linear trajectories may be used to replace traditional broadcast based discovery. If unicast communication is modeled by a simple curve, multicast is modeled by a tree in which each portion might be a curve, or a simple line. Distribution trees are used for either broadcasting (figure 5e), or multicast routing (figure 5f). A source knowing the area to be flooded can generate a tree describing all the lines to be followed by packets in order to achieve complete distribution with minimal broadcast communication overlap. A multicast source knowing positions for all members of a group may generate a spanning tree built of linear trajectories to be followed by packets. There is an overhead to be paid in describing the tree in each packet, but the solution saves in route maintenance.

4.1 Unicast routing

The prime application of forwarding is routing. The difference between the two is that in forwarding, a trajectory need not have a particular destination. Routing involves not only delivery to the destination, but the entire process that supports the delivery. This includes forwarding, and also building or updating routing tables. In order to route, the position of a given destination node is needed, as provided by a location service [20, 22], to enable node centric applications run on top of position centric routing. The other central problem is how to determine the actual trajectory. So far we experimented with simple trajectories, such as lines and sine waves, but the more general question is how to determine the trajectory when the position of the destination is known. If the topology is uniform in terms of density and capacity, it is likely that simple lines and parabolas for alternate paths would suffice. If the network exhibits more variation, in general shape of policy defined islands (of security, or capabilities for example), determination of the trajectory cannot be done with localized decisions. For these cases, we

Figure 7: Naive TBF based broadcasting



intend to explore the creation of a service responsible for trajectory mapping, that would take into consideration all the aspects in the creation of routes.

4.2 Multipath routing

More advantages are brought by TBF for multipath routing, which may be employed by a source to increase bandwidth or resilience of communication. The key feature here is the **cheap path diversity**. Using TBF, the source may generate either disjoint paths as disjoint curves, or braided paths as two intersecting sine waves. In networks with low duty cycles, such as sensor networks, longer alternate paths might actually be more desirable in order to increase the resilience of the transmitted messages (concept similar to Fourier decomposition), or to distribute the load onto the batteries. Since there is essentially no route maintenance, each packet can take a different trajectory, depending on its resilience requirements (similar to different FEC requirements). The multiple paths between a source and a destination can therefore be alternated to cheaply achieve load balancing.

4.3 Mobility

Mobile networks are a case in which TBF provides a desirable solution due to its **decoupling of path name from the path itself**. In a mobile ad hoc network, route maintenance for trajectory based routing comes for free since all that is needed is the position of the destination. This is especially true when only the intermediate nodes or the source are moving, and the destination of the packet remains fixed. When the destination is moving, a location service [20] may be used, or the source may quantify its uncertainty about the destination by using a localized flooding around the destination (figure 5e).

4.4 Discovery

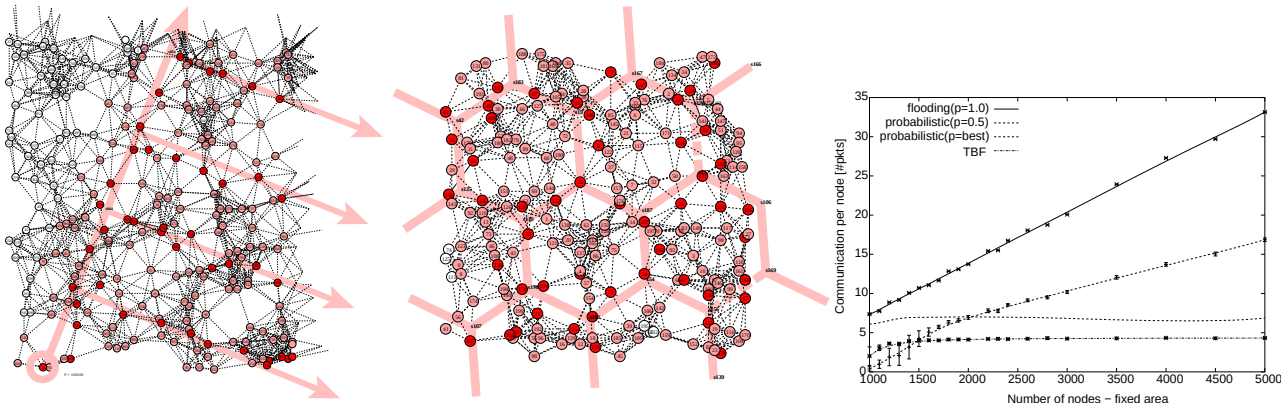
One of the areas in which TBF is particularly appropriate is **quick and dirty implementation of services** without the support of preset infrastructure. Such is the case of discovery - of topology, or of some resource. Many algorithms use initial discovery phases based on flooding [17, 5] in order to find a resource or a destination. Generalizing an idea presented in [27], a replacement scheme using trajectories is as follows: possible destinations (servers S) advertise their position along arbitrary lines and clients C will replace their flooding phase with a query along another arbitrary line which will eventually intersect the desired destination's line. The intersection node then notifies the client about the angle correction needed to contact the server directly (figures 5d and 6). In order to guarantee that the server and client lines intersect inside the circle with diameter CS, it is in fact necessary for the nodes each to send in four cardinal directions.

4.5 Broadcasting

Broadcasting is one of the most used primitives in any network, used for tasks ranging from route discovery at the network layer, to querying and resource discovery at the application layer. Its most frequent implementation is under the form of suppressed flooding, which entails each node of the network broadcasting the message exactly once. It is a **stateful** method, since it requires bits to mark the status of a node - covered or uncovered. The problem with the marking bits is that they have to be provisioned on a per broadcast basis, if several broadcasts are to be supported simultaneously. If only $O(1)$ marking bits are used, some global serialization is necessary. For example if one bit is used, one broadcast is supported in the network, and after the settling time (time at which last copy of a message is broadcast), the bit has to be cleared to allow for another broadcast. Suppressed flooding also incurs several other problems: increased communication [28], increased settling, time poor scalability and delivery ratio in congested networks [29]. Probabilistic flooding [30] addresses some of these problems by flipping a coin each time a node has to rebroadcast the message. This reduces the number of duplicates a node receives, but the method exhibits a bimodal behavior, meaning that either the broadcast is successful in covering most of the network, or it dies very early, covering only a small portion around the source. While broadcasting is not the main application of TBF, we can provide solutions that address most shortcoming of traditional flooding and of probabilistic flooding. The broadcast achieved by TBF also has an approximate nature, just like probabilistic flooding, meaning that there may be nodes which do not receive the message even under ideal collision free conditions.

We discuss here two classes of broadcast approximation: stateless and stateful. Stateful broadcasting 8(b) is similar in nature to classical flooding in that it requires per source marking bits in order to suppress duplicates. State-

Figure 8: TBF based broadcasting



(a) parallel (stateless) (b) honeycomb (stateful) (c) Performance

less broadcasting (figures 7a and 8a) has the property that no memory is used at nodes in order to mark the visited/non visited status. The broadcasting scheme is designed to not overlap itself indefinitely, although it may intersect itself. Settling time is therefore less of a factor, since different broadcasts need not choose between serial execution and the use of marking bits. They can overlap without interference problems at the algorithmic level. As an example of naive stateless broadcasting, in figure 7(a), the node in the upper left corner initiates radial spokes that cover directly the nodes involved in forwarding, and indirectly nodes which are at most one hop away from the trajectory. The performance curves in figure 7(b) are for an initiator node in the middle of the network. As the number of spokes used increases, the coverage increases, but also does the communication spent. When seen as a fraction of communication spent by classical suppressed flooding, the naive method performs remarkably well achieving 95% coverage with less than half the communication. However, for larger networks, spokes will diverge, leaving large areas uncovered.

In figure 8 we investigate alternative broadcasting schemes. The parallel structure of the spokes proves to be quite economical in terms of overhead because spokes spaced at about twice the communication range tend to reduce the number of duplicate packets, while providing high coverage. Both stateless schemes - the radial and the parallel spokes - have a lower predictability since large portions of the network may remain uncovered due to a break in the trajectory.

For stateful schemes, we experimented with three plane filling patterns - triangles, squares and hexagons (figure 8(b), the honeycomb structure is also explored in [31], under the name of "optimal flooding"). These scheme resemble classical suppressed flooding with respect to coverage and predictability, but still have the property of limiting the amount of duplicates specific to all TBF based methods. They presented similar performance under ideal conditions, although a square shape seems to be a little better, however not as good as the parallel spoke stateless scheme.

In figure 8(c), we compare the number of packets handled per node among three routing schemes: classic, probabilistic, and TBF based with the square pattern (honeycomb and triangular are slightly higher, while stateless parallel is slightly lower in communication). As we increase the number of nodes in a fixed area, classical flooding's communication increases linearly (with the average number of neighbors), but also does the probabilistic flavor after a certain density. The problem is that for each density, a different probability is appropriate, and by choosing it manually it is actually possible to obtain constant communication cost per node - indicated by the curve labeled "p=best" in the figure. Aside from the actual performance, it is worth noting that probabilistic broadcasting has a bimodal nature, visible in the high variance for low densities, and better documented elsewhere [30].

The results show that broadcasting along predefined plane filling lines performs very well with increasing density in a fixed area, because the number of transmissions only depends on the fixed area to be covered, and not on the actual number of nodes receiving the packet. The main advantage stems from the fact that a node receives a packet only about three or four times, instead of receiving it from all of its neighbors, as in classical suppressed flooding.

It is worth noting that plane filling patterns also have applications in topology discovery: if a node wants a low resolution image of the network, without the burden of querying for the entire detailed topology, it may employ a pattern like the honeycomb, but with a larger size. This will have the flowing property of flooding to wrap around obstacles and go through bottlenecks in the topology, but will touch only a fraction of the nodes. It is true that obstacles and bottlenecks have to be larger than the pattern size in order to achieve full coverage, but the result is a finely tunable tradeoff between communication spent and resolution obtained.

4.6 Multicast

Another example of quick and dirty implementation of a network service is multicast. Traditional multicast implementation is overkill in many sensor or ad hoc networks applications, because of group setup and tree maintenance. There are situations when only one shot multicast is necessary, to send a query or a notification, and a tree setup would be unjustified. Even when the membership has a

Figure 9: Loop method

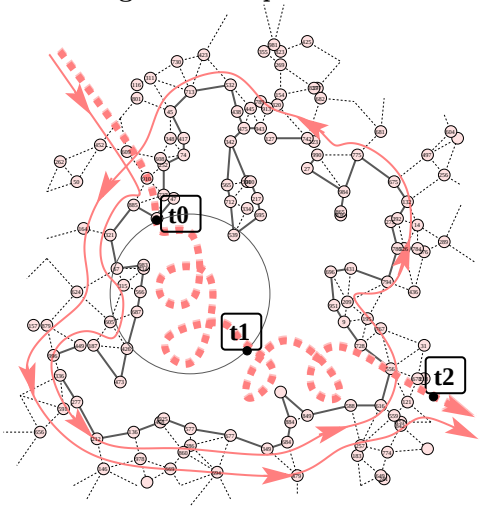
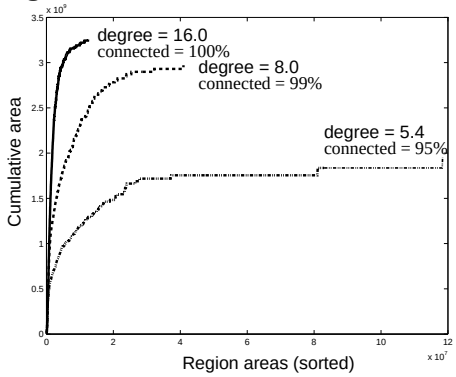


Figure 10: Obstacle area distribution



longer duration, mobility could render the connecting tree useless. For such cases, the source node, assuming it has the positions of all the receivers, can determine an approximate Euclidean Steiner tree to be used as a distribution tree, without using other communication except position notification from the receivers.

5. ADVERSE CONDITIONS

The question we address in this section is "how does TBF perform when the initially stated conditions are not met?". We investigate here the aspects of reduced density and reduced or inexistent positioning support infrastructure. A sparse network can be seen either as containing obstacles that may block trajectories, or more generally as policy islands that determine routing decisions. A trajectory that intersects such an obstacle has the option of simply dropping the packet, or routing around the obstacle.

Positioning hardware may not be desirable on each node due to energy, cost, form factor, or service availability reasons. If some fraction of the nodes have self positioning capability, approximate positioning in a global coordinate system may be established [8, 9]. If no self positioning capability is available, TBF can be supported by a localized scheme that positions only nodes along the trajectory [11].

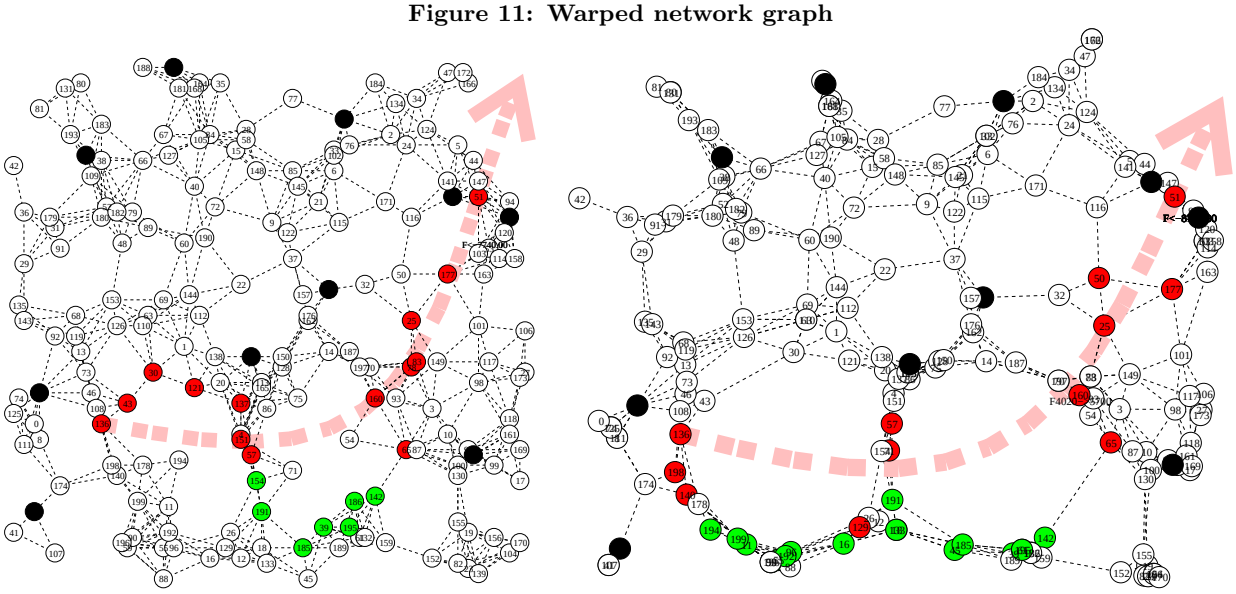
5.1 Sparse networks

Dealing with the sparseness of the network addresses in fact a larger class of problems: obstacles, low connectivity areas, dead or sleeping nodes, policy defined islands that are to be avoided – islands of low energy or vulnerability. In TBF, a trajectory is most of the time advanced by greedy decisions (section 3.1) taken during forwarding. To find a path around the obstacle for cartesian forwarding [6], localized methods explored in the literature include flooding and depth first search. These however require maintenance of per destination states in the forwarding nodes, and extra communication, posing scalability problems. The most relevant work related to avoiding obstacles in routing is the FACE algorithm [14](see also [15]), which is stateless and localized. It operates on planar graphs only, so the first step involves localized elimination of the edges that are not part of a known planar graph, such as relative neighborhood graph, Gabriel graph, or Delaunay graph. Routing is then done in one of the two modes: greedy, choosing the node closest to the destination (MFR[26]), or FACE mode. The latter uses the "right hand rule", which involves following one side of the polygon, until finding a edge that intersects the source destination line, at which point greedy forwarding mode is resumed. While the online nature of this algorithm is desirable, as it allows for localized decisions, the algorithm cannot be trivially adapted to work with the general trajectories. An intuitive argument is that inside the obstacle the curve may oscillate back and forth, regardless of the advance made by FACE packet on the border, making it harder to detect the exit point. Detecting the exit point for a straight line trajectory involves just the intersection of a line and a segment, whereas for arbitrary trajectories it requires equation solving using numerical methods.

A simple solution would be to premap all the polygons determined by the planarization, and store each polygon in all the border nodes it is made of. When a trajectory arrives at a node where greedy TBF is not possible, the node would have all its adjacent polygons stored locally, thus being able to plot a path around the offending polygon, using the FACE strategy. This preprocessing method would only be appropriate for static networks and when polygon sizes are small. For example, nodes on the outer face would have to store the entire perimeter of the network, which may be prohibitive in scale.

What we propose eliminates the need of preprocessing, but may incur either extra computation or extra communication. It requires the sender of the trajectory to have a rough estimate of the size of the obstacles, and of the possible behavior of the curve inside such sized obstacles. Associated with the trajectory, the sender attaches Δ – an estimated diameter of the obstacles in terms of t. In figure 9, the dashed arrow shows the desired trajectory, while the continuous one shows the trajectory achieved by the obstacle avoidance method. When greedy mode is no longer possible at t_0 on the curve, the responsible node switches to FACE mode and continues forwarding along the border using the "right hand rule", shown in the picture with a continuous arrow. The problem is that each side of the polygon must be tested for a possible exit point of the curve. Each node forwarding in FACE mode evaluates the curve from the point

Figure 11: Warped network graph



t_0 to the point $t_1 = t_0 + \Delta$, and switches back to greedy mode if it is the case. If Δ is underestimated, as in figure 9, FACE mode will loop around the obstacle reaching point t_0 without finding an exit point. In a second tour of the polygon, all nodes evaluate the curve between $t_1 = t_0 + \Delta$ and $t_2 = t_0 + 2\Delta$, with a node being successful in finding the exit point. A large underestimation of Δ leads to wasted communication, as in the example here. Overestimation leads to high computational strain on the border nodes, as they have to compute a large portion of the curve that is outside the obstacle they are a part of. A way to limit the amount of curve evaluations is to send along with the FACE packet a circle enclosing the evaluated $[t_0 + i\Delta,\ t_0 + (i+1)\Delta]$ portion of the curve. This circle is computed by the node at t_0 and enlarged slightly to ensure that segments on the obstacle perimeter having both ends outside the circle need not evaluate the Δ portion of the curve.

The method is a good engineering compromise for obstacle avoidance if the approximate diameter of the obstacle is known statistically. Its advantages are that it requires no state in the border nodes, and no preprocessing, thus being appropriate for mobile scenarios. In networks with a uniform distribution of nodes, it turns out that when there is enough density to guarantee connectivity, the uncovered areas (obstacles) are fairly small and with predictable maximum size. In figure 10, unit graph networks of three different densities are planarized by retaining only the edges in Gabriel graph, and the subdivisions resulted are sorted by area. For low densities, there are two noticeable aspects: the cumulative of all subdivisions barely covers half of the network, and there is a large number of large area regions. Higher densities show a distribution of obstacle size that is more friendly to TBF: most of the total area is covered by small regions, and there may be little need for using FACE mode. This means that in uniform density networks, we can have a reasonable estimate for Δ, that would avoid excessive looping or excessive curve evaluations.

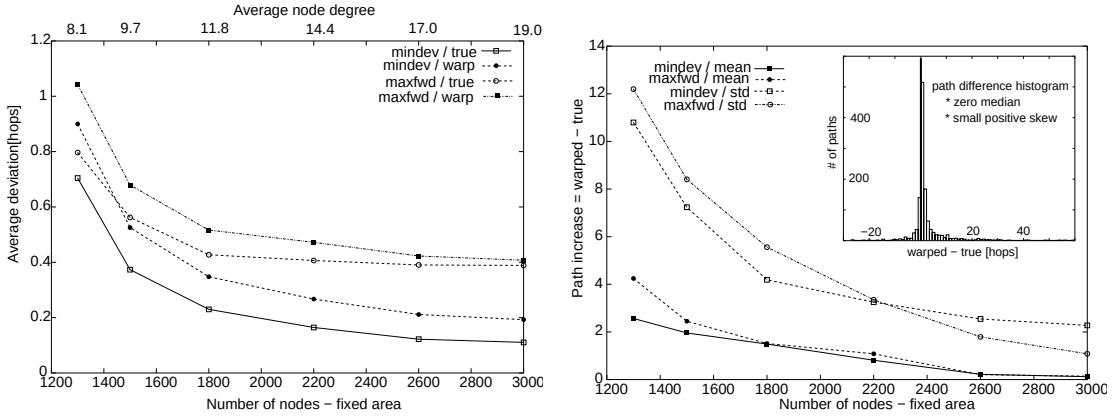
5.2 Imprecise locations

An implicit assumption we have made so far, and one that is made by most position centric schemes is that if node positions are available, they are perfect. This assumption is not always valid, and not only due to positioning device accuracy. In the event that GPS is not available throughout the network, it is possible to either agree on some relative coordinate system [32], or have a small fraction of GPS enabled nodes disseminate location to the rest of the network [8, 9, 10]. Most of these schemes employ ranging methods based on signal strength, ultrasound, angle of arrival, or just plain hop distance. All these provide estimates with some amount of error, which is finally translated in positioning error. In this section we explore the behavior of TBF under the assumption that node positions are approximative, as they are provided by a positioning algorithm. For this purpose we used DV-hop [8], a method that only uses hop distances to estimate positions. Its basic idea is that when a node knows shortest distances in hops to three different known landmarks, it can infer its position. This is done by approximating euclidean distance using hop distances, and assuming that the exact positions of the landmarks are known. DV-hop does not require additional capabilities, such as ranging, or AoA, but only produces acceptable positions in isotropic[1] networks.

As we will show, when using approximate (warped) positions, TBF incurs only a small penalty in efficiency, thus being usable even when not all nodes are GPS equipped. The networks that we consider for this experiment are all isotropic, with uniform distribution of nodes, and with increasing densities obtained by increasing the number of nodes in a fixed area, with a fixed communication radius. The position error, or distance from the true location, is on average one half the communication range, but may be larger for fringe nodes, due to the well known edge effects. The central aspect of this setup is that although positions are affected by

[1] *isotropic* = having the same physical properties in all directions (connectivity, density, node degree)

Figure 12: TBF on DV-hop



(a) Deviation from the ideal trajectory

(b) Path length increase

errors, the resulted image of the network is coherent. In figure 11, a network with 200 nodes 10 of which are landmarks, is shown with true locations on the left, and with warped locations obtained by DV-hop, on the right. The warped graph, although not a unit disk graph, like the original, has the same set of nodes and edges like the original. It can also also be planarized, for the purposes of guaranteed delivery [14]. The problem is that in the warped space two nodes can be very close and not have a link between them, or be far apart and have the link. This prevents the use of localized planarization procedures available for planar subgraphs such as Gabriel graph or relative neighborhood graph. However, for the rest of this section, we assume that the graph is planarized properly, in order to use the obstacle avoiding strategies outlined in section 5.1. Given that the trajectory is expressed in the warped coordinate system, on which all nodes implicitly agree, forwarding can then be performed just like in any planar subdivision.

While the actual forwarding path will always deviate from the ideal trajectory (when density is finite), even with perfect positions, we want to see how this deviation is affected by the warped positions provided by DV-hop. In order to quantify this drift, in networks with 1300-3000 nodes, we randomly chose 200 pairs of nodes and forwarded along linear trajectories in the two spaces. The paths had on average 15-25 hops, depending on the density. For TBF forwarding, two policies were considered - MFR and "minimum deviation", explained in section 3.1. The main parameter of these simulations was density, as we wanted to capture the behavior in both avoiding obstacles (low densities), and greedy forwarding (higher densities).

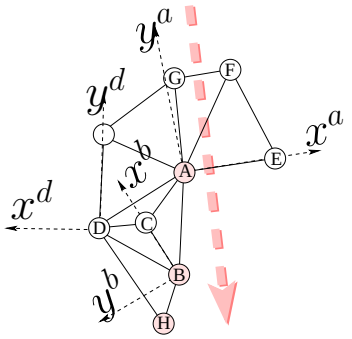
In figure 12a, average deviation from the ideal parametric trajectory for both the real and the warped networks is under one communication radius for all but the lowest density. The deviation is computed as the length of the perpendicular from the actual node used in forwarding to the ideal trajectory in the true space, whether in greedy or in FACE mode. As we expect, deviation decreases with density, and "minimum deviation" sticks closer to the trajectory than MFR. The unexpected result is the small difference in deviation between routing with true positions and with warped positions. In figure 12b, we examine the difference in the length of the path in hops between the warped network and the real network. As the inset histogram shows, this difference is not always positive, that is, the path in the warped network is not always longer. In fact, the distribution of this difference has a zero median, and a small positive mean with a large standard deviation. Because obstacle avoidance decisions are localized, certain decisions in forwarding might incur or avoid large detours. The intuition behind this behavior is that certain obstacles may be avoided in the warped network, and not in the real one, or vice-versa.

Overall, the performance in terms of deviation and path length is only slightly deteriorated by the use of imprecise positions, which shows that TBF is robust in the face of realistic corruptions in node positioning. Investigation of alternative positioning schemes and efficient planarization of the warped graph remain interesting open issues.

5.3 Alternatives to positioning

Trajectory based forwarding is simpler when all node positions are known relative to a reference coordinate system by use of a global capability such as GPS, or running a self positioning algorithm such as [8, 9, 33, 34]. In many cases however, running a network wide positioning algorithm is expensive and unnecessary. In this section, we shortly review LPS [11], a method that enables TBF even when such a global or ad hoc capability is not available due to place of deployment (e.g. indoors), high communication costs, or additional infrastructure requirements [24, 35, 33]. With this method, nodes use some other capabilities (ranging, AOA, compasses) to establish local coordinate systems in which all immediate neighbors are placed. It is then possible to register all these coordinate systems with the coordinate system of the source of the packet. Local Positioning System (LPS) is a method to achieve positioning **only for the nodes along the trajectory**, with no increase in communication cost, as if all node positions were known. Each node touched by the trajectory will spend some computation to position itself in the coordinate system of the source of the packet,

Figure 13: Local coordinate systems



trading off some accuracy of the trajectory.

In figure 13, each node, using some local capabilities sets up a local coordinate system. Registration between two local coordinate systems is the process that computes a transformation which will translate any point from one coordinate system to the other. The input to this process are points for which the coordinates are known in both coordinate systems with some accuracy.

The aim for LPS is to make forwarding along the trajectory similar to the procedure followed in a network where node positions are available. The key idea is that the only nodes that are positioned are the ones involved in forwarding along the trajectory. Positioning is done in a hop by hop fashion, in the coordinate system chosen by the initiating node - the source of the packet.

The forwarding procedure works with a node selecting the next hop based on the proximity to the desired trajectory, or any of the other possible policies. In figure 13, the ideal trajectory is shown as a thick dashed arrow. Assume that A knows the equation of the trajectory in its own coordinate system, which has been already registered to the coordinate system of the packet. If the next node to be selected along the trajectory is B, it will receive from A all the necessary data to register its own coordinate system to A's by solving a localized optimization problem. Node B is then able to select one of its own neighbors that is closer to the trajectory, in order to continue the process.

What is in fact achieved by LPS is the registration of all coordinate systems of visited nodes to the coordinate system of the initiating node, which achieves positioning of all these nodes in the coordinate system of the source. The method is localized to the nodes actually touched by the trajectory. Unlike a network wide positioning algorithm, such as [8, 9], which involves collaboration and coordination of a large number of nodes, LPS involves only the nodes "touched" by the desired trajectory.

According with the results presented in [11], LPS based forwarding maintains a low deviation, making it usable for discovery purposes, and reaches the destination with high probability, making it usable for routing. Although error compounds with distance, it can be countered efficiently by using additional hardware available in small form factors. An accelerometer present in each node detects node flipping, eliminating false mirroring, while a digital compass eliminates rotations from registrations process, when angular measurements are used.

6. RESEARCH ISSUES

Currently, TBF has been extensively simulated to verify most of the hypotheses presented here, like its use for routing, discovery and broadcasting, as well as adaptation under adverse conditions, like the lack of GPS, and forwarding in sparse networks. In addition, TBF has also been implemented on a Mica mote sensor testbed, and preliminary experiments with basic trajectories are under way. Several issues related to determining, specifying and modifying the trajectory are still under consideration. While any trajectory in the plane can theoretically be expressed as a parametric curve, many applications may require description of large trees, or of large parametric expressions. Here we intend to explore efficient representation methods based on lossy compression methods such as Fourier, or wavelets. Concepts from active networking might also be viable in sending a function in a compact compiled form, especially for the recursive plane filling patterns. Another issue is patching and modification of the trajectory. Modification can be decided by intermediate nodes in order to impose local detours, without involving all the upstream or downstream nodes. Trajectory can also be modified globally if better information about the destination becomes available.

In many cases, the trajectory is not available in parametric form at the source, but as a list of points. Depending on the requirements of the application and the pattern described by the curve, the possibilities for curve encoding range from an anchor list of all points (which is yet another form of source based routing, with cartesian forwarding between intermediate points), to a curve fitting using a function base (Fourier or polynomial). Curve fitting is attractive in cases where the number of anchors is large and the requirement of touching all anchors is not strict.

Although TBF is characterized as a generalization of source based routing, it is not clear who should specify the actual trajectory. For example, if a sink is collecting data from several sources, it can specify trajectories that would achieve optimal aggregation and minimum overall communication, based on information which is not available at each individual source. Also, when a destination has more information about the network, it might provide better routing around obstacles, or a single route that would serve multiple close destinations.

TBF can be used as a low level primitive in implementing many network protocols in ad hoc networks. In static networks it can be used for end to end routing, either coupled with a discovery phase, or with a location service. In mobile networks, TBF is immune to source mobility and intermediate node mobility, since the trajectory does not explicitly encode intermediate members of the path. The research issue remains the mobility of the destination, and a possible solution could involve a strategy based on TBF and DREAM [21].

7. CONCLUSIONS

We presented Trajectory Based Forwarding (TBF), a novel paradigm that makes the transition from a discrete view of the paths to a continuous view of the paths in future dense networks. The method is a hybrid between source based routing and cartesian forwarding in that the trajectory is set by the source, but the forwarding decision is local and greedy. Its main advantages are that it provides cheap path

diversity, decouples path naming from the path itself, and trades off communication for computation. When GPS is not available, TBF may make use of alternate techniques, such as global and local positioning algorithms. It is robust in front of adverse conditions, such as sparse networks, and imprecise positioning. We believe that TBF should be used as an essential layer in position centric ad hoc networks, as a support for basic services: routing (unicast, multicast, multipath), broadcasting and discovery.

8. ACKNOWLEDGMENTS

This research work was supported in part by DARPA under contract number N-666001-00-1-8953 and NSF grant ANI-0240383.

9. REFERENCES

[1] Brett Warneke, Matt Last, Brian Leibowitz, and Kristofer Pister. Smart dust: Communicating with a cubic-millimeter computer. *IEEE Computer*, 34(1):45–51, January 2001.

[2] Tomasz Imielinski and Samir Goel. Dataspace - querying and monitoring deeply networked collections in physical space. *IEEE Personal Communications Magazine*, October 2000.

[3] Tomasz Imielinski and Badri Nath. Wireless graffiti: data data everywhere. In *VLDB, Invited paper, 10 year VLDB-award*, Hong Kong, August 2002.

[4] Vladimir Lumelsky, Michael Shur, and Sigurd Wagner. Sensitive skin. *IEEE Sensors Journal*, 1(1):41–51, June 2001.

[5] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *ACM MOBICOM*, Boston, MA, August 2000.

[6] G. Finn. Routing and addressing problems in large metropolitan-scale internetworks. Technical Report ISI Research Report ISI/RR-87-180, University of Southern California, March 1987.

[7] Tim Roughgarden and Éva Tardos. How bad is selfish routing? *Journal of the ACM (JACM)*, 49(2):236–259, 2002.

[8] Dragoş Niculescu and Badri Nath. Ad hoc positioning system (APS). In *GLOBECOM*, San Antonio, November 2001.

[9] A. Savvides, C.-C. Han, and M. Srivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *ACM MOBICOM*, Rome, Italy, 2001.

[10] Dragoş Niculescu and Badri Nath. Ad hoc positioning system (APS) using AoA. In *INFOCOM*, San Francisco, CA, April 2003.

[11] Dragoş Niculescu and Badri Nath. Localized positioning in ad hoc networks. In *Sensor Network Protocols and Applications*, Anchorage, Alaska, April 2003.

[12] J. C. Navas and Tomasz Imielinski. Geographic addressing and routing. In *ACM MOBICOM*, Budapest, Hungary, September 26-30 1997.

[13] Y.-B. Ko and N. H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *ACM MOBICOM*, October 1998.

[14] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In *3rd International Workshop on Discrete Algorithms and methods for mobile computing and communications*, Seattle, WA, August 1999.

[15] B. Karp and H.T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *ACM MOBICOM*, Boston, MA, August 2000.

[16] Fabian Kuhn, Roger Wattenhofer, Yan Zhang, and Aaron Zollinger. Geometric ad-hoc routing: Of theory and practice. In *22nd ACM Symposium on the Principles of Distributed Computing (PODC)*, Boston, MA, USA, July 2003.

[17] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. *Mobile Computing, Kluwer Academic Publishers*, 353, 1996.

[18] Charles E. Perkins. *Ad Hoc Networking*. Addison Wesley Professional, 2001.

[19] Ljubica Blazevic, Silvia Giordano, and Jean-Yves Le Boudec. Self organized terminode routing. In *Cluster Computing*, volume 5, pages 205–218, 2002.

[20] J. Li, J. Jannotti, D. De Couto, D. Karger, and R. Morris. A scalable location service for geographic ad hoc routing. In *ACM MOBICOM*, pages 120–130, Boston, MA, August 2000.

[21] Stefano Basagni, Imrich Chlamtac, Violet R. Syrotiuk, and Barry A. Woodward. A distance routing effect algorithm for mobility (DREAM). In *ACM MOBICOM*, pages 76–84, Dallas, TX, October 1998.

[22] P. H. Hsiao. Geographical region summary service for geographical routing. *Mobile Computing and Communication Review*, 5(4):25–39, January 2002.

[23] Deepak Ganesan, Ramesh Govindan, Scott Shenker, and Deborah Estrin. Highly resilient, energy efficient multipath routing in wireless sensor networks. In *Mobile Computing and Communications Review (MC2R)*, volume 1, 2002.

[24] B.W. Parkinson and J.J. Spilker. *Global Positioning System: Theory and Application*. American Institute of Astronautics and Aeronautics, 1996.

[25] Chris Savarese, Jan Rabaey, and Koen Langendoen. Robust positioning algorithms for distributed ad-hoc wireless sensor networks. Technical report, Delft University of Technology, 2001.

[26] Leonard Kleinrock and John Silvester. Optimum transmission radii for packet radio networks or why six is a magic number. In *IEEE National Telecommunications Conference*, pages 4.3.1–4.3.5, Birmingham, Alabama, 1978.

[27] Ivan Stojmenovic. A scalable quorum based location update scheme for routing in ad hoc wireless networks. Technical Report TR-99-09, SITE, University of Ottawa, September 1999.

[28] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. In *ACM MOBICOM*, pages 151–162. ACM Press, 1999.

[29] B. Williams and T. Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pages 194–205, 2002.

[30] Zygmunt Haas, Joseph Halpern, and Li Li. Gossip

based ad hoc routing. In *INFOCOM*, New York, USA, June 2002.

[31] Vamsi S. Paruchuri, Arjan Durresi, Durga S. Dash, and Raj Jain. Optimal flooding protocol for routing in ad-hoc networks. Technical report, Ohio State University, CS Department, 2002.

[32] S. Capkun, M. Hamdi, and J.-P. Hubaux. GPS-free positioning in mobile ad-hoc networks. In *Hawaii International Conference On System Sciences*, Outrigger Wailea Resort, January 3-6 2001. HICSS-34.

[33] N.B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *ACM MOBICOM*, Boston, MA, August 2000.

[34] Paramvir Bahl and Venkata N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *INFOCOM*, Tel Aviv, Israel, March 2000.

[35] Nirupama Bulusu, John Heidemann, and Deborah Estrin. GPS-less low cost outdoor localization for very small devices. In *IEEE Personal Communications Magazine*, Special Issue on Smart Spaces and Environments. October 2000.