Efficient route discovery in hybrid networks

Roy Friedman, Ari Shotland, Gwendal Simon

Efficient route discovery in hybrid networks ☆

Roy Friedman [a], Ari Shotland [a], Gwendal Simon [b,*,1]

[a] Technion Israel Institute of Technology, Computer Science Department, Haifa 32000, Israel
[b] Institut TELECOM, TELECOM Bretagne, Computer Science Department, Brest 29238, France

A B S T R A C T

Hybrid networks are formed by a combination of access points and mobile nodes such that the mobile nodes can communicate both through the access points and using ad-hoc networking among themselves. This work deals with providing efficient routing between mobile devices in hybrid networks. Specifically, we assume the existence of a spanning tree from each access point to all mobile devices within the transitive transmission range of the access point. We utilize this spanning tree to design a family of efficient point-to-point routing protocols for communication between the mobile devices themselves. The protocols utilize the tree structure in order to avoid expensive flooding of the entire network. The paper includes a detailed simulation study of several representative communication patterns, which compares our approaches to DSR.

Keywords:
Hybrid networks
Access points
Ad-hoc networks
Route discovery

1. Introduction

1.1. Motivation

Combining ad-hoc networking mechanisms with access points offers an interesting prospective for mobile computing. From the point of view of mobile networks operators, such hybrid networks can be used to increase the coverage area offered by the network at no additional cost to the operator. Also, communication between nodes that are within the vicinity of each other can be carried through the ad-hoc networking mechanism, thereby reducing the traffic on the operator's access point. This can be used, e.g., for collaborative applications and caching [20]. Moreover, results from studies of networks capacity suggest that, capacity wise, ad-hoc networks are not very scalable [8,18,21,23,24], at least when there are no restrictions on the mobility model and on communication patterns. On the other hand, by combining a sufficient number of access points with ad-hoc networking, the capacity of the network can remain constant for any network size [4,27,48].

When considering the ad-hoc network part, we speculate that a large fraction of the communication will be from mobile devices to access points and vice versa. This suggests that it is beneficial to maintain paths between each mobile device and the access point(s) in its vicinity in a proactive manner. Yet, maintaining an independent path from each mobile device to its nearby access points is not scalable either. This is because of the heavy load associated with discovering routing paths in ad-hoc networks and the cost of periodic maintenance of paths.

Thus, instead, it makes sense to construct and maintain a spanning tree from every access point to all mobile devices in its vicinity [41]. Many recent wireless protocols, including ZigBee [2], consider routing through such spanning trees, also known as hierarchical routing. These spanning trees define an *extended coverage area* for each access point. Yet, given the collaborative nature of the applications we mentioned above, there is a fair chance that some nodes within the same extended coverage area may wish to communicate with each other. An interesting question

☆ This work is funded by France Telecom R&D.
* Corresponding author. Tel.: +33 229 001 573; fax: +33 229 001 282.
E-mail addresses: roy@cs.technion.ac.il (R. Friedman), ari@cs.technion.ac.il (A. Shotland), gwendal.simon@telecom-bretagne.eu (G. Simon).
[1] Most of this work was done while this author was with France Telecom R&D.

is how to utilize the assumed spanning trees in order to facilitate efficient point-to-point communication between mobile nodes within the same extended coverage area. This question is the focus of this paper.

1.2. Contribution of this work

In this work we investigate how to improve the routing costs of messages whenever a spanning tree exists,[2] compared to using a standard routing protocol like DSR [32]. Specifically, we start by exploring the communication costs of DSR compared to a simple protocol, which we call ST, that always routes messages along the edges of the spanning tree. As we show from our empirical study, when a path is only utilized a small number of times, ST involves a much smaller communication overhead than DSR. However, as the frequency of using the path increases, DSR becomes better. This can be explained by the fact that DSR pays a high price for finding the shortest path between a source s and a destination d, regardless of the frequency in which the path will be used. However, once this path is found, all messages between s and d are routed on this path, whose cost is minimal. On the other hand, ST spends very little overhead on finding routing paths, and so its cost comes from forwarding messages in a sub-optimal path. Thus, DSR has an initial high cost and a small gradual cost, while ST has a zero initial cost and a high gradual cost (this is under the assumption that the spanning tree is maintained in any case due to the hybrid nature of the system, as discussed above).

These observations led us to look for protocols whose cost is similar to ST when a path is rarely used, yet if the communication between a pair of nodes becomes popular, then they will behave similarly to DSR. That is, our goal is to find and investigate spanning tree-based protocols that employ gradual path learning, which have the following appealing properties:

(1) The amount of effort to find a path between two nodes depends on its popularity.
(2) The length of the routing path between any two nodes p and q monotonically improves with its utilization. It is always between the length of the path that includes the common ancestor of both p and q in the tree and the shortest path from p to q in the real network. If the path is used often enough, it will eventually converge to the shortest path (or very close to it).
(3) The protocol avoids flooding the entire network.

We assume that every node p in the tree can tell whether another node q is in any of the sub-trees rooted by each of p's children in the tree. In ZigBee, the way the network identifiers are delivered aims to achieve this assumption. In other protocols, it can be achieved, for example, by maintaining full knowledge of the nodes in each sub-tree, or through time and space efficient summaries like Bloom filters [12]. Nevertheless, we observe that this is obtained by a simple mechanism in which, starting from the leaves, each node propagates the information about all of its children to its parent in the tree.

In addition to introducing these intermediate protocols, our paper also includes an extensive evaluation by simulations. In these simulations, we compare our protocols to each other as well as to the well known DSR point-to-point routing protocol [32]. The results show that indeed, for each communication pattern investigated, some of our protocols behave better than DSR and ST.[3] Finally, we also present an on-line competitive analysis in which we compare some of our protocols with an optimal algorithm that always knows ahead of time how often two nodes will communicate.

The rest of this paper is organized as follows: Related works are discussed in Section 2. The basic model assumptions are described in Section 3. Our family of protocols is introduced in Section 4. The simulation results appear in Section 5, while the competitive analysis is presented in Section 6. We conclude with a discussion in Section 7.

2. Related work

Routing techniques in MANET are often classified into *proactive*, e.g., Optimized Link-State Routing (OLSR) [15] and Topology-Based Reverse Path Forwarding (TBRPF), *reactive*, e.g., Ad-hoc On-demand Distance Vector (AODV) [42], and Dynamic Source Routing (DSR) [32], and a mixture of both, e.g., ZRP [26], as well as geographic routing [34,36,40,43]. The use of trees for routing is advocated in several scenarios, including group communication [38] and energy efficient data gathering [29].

An interesting approach to reducing the cost of routing in ad-hoc networks is through the use of *clustering*. That is, the network is divided into clusters such that each cluster has a *clusterhead* [6,7,10,11,19]. For routing, each node forwards the message to its nearest clusterhead, who then uses a routing protocol to route the message to the clusterhead closest to the destination. This way, routing paths are only maintained between clusterheads. These paths are guaranteed by this scheme to be heavily used, and not change too often, which makes the cost of maintaining them worth it. Specific examples include [31,37,45].

A somewhat similar approach is to use minimal connected dominating sets (MCDS), e.g., [5,17,49,50]. A connected dominating set (CDS) is a connected set of nodes such that every node is either a member of the CDS, or a direct neighbor of a member. Thus, the nodes of the CDS form a backbone, or overlay, on which messages can be routed. Since it includes much fewer nodes and links than the entire network, finding routing paths along the CDS is more efficient than doing a flood search in the entire network.

[2] In this work, we do not address the construction and maintenance of such spanning trees, but rather rely on known techniques, e.g., [14,41], for that. However, our simulations take into account the cost of building such a tree.

[3] We have also compared with AODV [42], but in the scenarios we measured, the behavior of AODV was very similar to DSR, as both employ flooding, so we only report on DSR.

An important difference, however, between both clustering-based routing and MCDS-based routing and our approach, is the particular way that we use the overlay (in our case, the spanning tree). In our work, the emphasis is on using the tree only if the frequency of a given path usage does not merit finding the optimal route for it. Also, when looking for the optimal path, most of our proposed schemes try to utilize the tree to make the task of finding the shortest path more efficient, as detailed in Section 4.

As for hybrid networks, most of works published so far have integrated Mobile IP (MIP) with some ad-hoc routing protocol in order to allow for Internet connectivity for nodes in the ad-hoc network. For example, [39] combines a modified Routing Information Protocol (RIP) with MIP, while [13] uses DSR inside the ad-hoc network. Alternatively, in the MIPMANET solution [33], nodes in an ad hoc network that require Internet access register with a foreign agent and use their home address for all communication. Mobile nodes tunnel all packets destined for the Internet to their Mobile IP foreign agent. The AODV routing protocol is used to discover routes between mobile nodes and the foreign agent. A similar approach was also investigated in [47]. Our work is orthogonal to these works, as we are looking at improving the cost of routing between peers in the same ad-hoc segment of a hybrid network, by utilizing a spanning tree from the access point to the mobile devices.

The work reported in [46] considered two routing protocols for hybrid networks. The first assumed a default gateway, while the other was a reactive protocol. It was shown that each performs better based on the communication pattern of the applications. Two additional route discovery protocols for hybrid networks were reported in [28,44]. These protocols mainly focus on discovering a route between a given mobile node and the access point. Additionally, the DST protocol described in [30] maintains a spanning tree of the network for routing purposes. It can handle changes at a cost of $O(\log n)$ and was shown by simulations to perform well. Yet, in their work messages are only routed along the tree edges, regardless of path popularity.

Recent works related to the 802.15.4 wireless protocol [3], and all upper layer details addressed by ZigBee standard [2], considers both unicast routing, usually implemented by AODV, as well as hierarchical routing, which basically uses a tree [16,25]. These works focus on improving the ZigBee hierarchical routing. Yet, in ZigBee, the hierarchy is defined based on the addressing scheme of the network. Also, the schemes reported in papers like [16,25] do not try to utilize the number of times a route is chosen in order to decide between AODV and hierarchical routing, and none of them attempts to shortcut the tree.

The work of Badia et al. investigated maintaining QoS levels in a heterogenous ad-hoc network [9]. They have considered a combination of proactive routing with a QoS-driven control mechanism for ensuring high bandwidth, low latency, and improved energy efficiency. Our work is complementary to such ideas, since we focus on improving the cost of routing inside a single tree.

3. Model

3.1. Communication model

We consider a set of mobile devices having wireless communication capabilities, and in particular, equipped with an omnidirectional antenna. Multi-hop ad-hoc routing is possible and we shall use the terms *path* and *route* interchangeably. For the purpose of the formal model, we assume a simplified *disk transmission model* [23].[4] That is, we assume a *transmission range* $\mathscr{R}$ such that a node q can receive messages transmitted by node p only if the distance between p and q is at most $\mathscr{R}$. Moreover, when this distance is less than $\mathscr{R}$, then q can receive any message sent by p (unless a third node within $\mathscr{R}$ distance from q transmits a message at the same time as p).

3.2. Spanning trees

As mentioned in the Introduction, we assume that nodes are organized in spanning trees. For the purpose of this work, we only address communication among nodes that share the same spanning tree. The exact protocol for building the spanning tree is outside the scope of this paper, but see [14,41]. Additionally, we assume that with high probability that the distance in the tree between any two nodes x and y that are within transmission range of each other is at most two and the difference between the tree depth of x and that of y is at most one level (in the tree). These assumptions are satisfied by any "hello"-based spanning tree construction protocol, such as [14]. Furthermore, the tree protocol is adaptive to changes in the network. We do not impose any restrictions on the distributions of the nodes in the network, yet some of our protocols produce much better results when the distribution is uniform.

Finally, as mentioned in the Introduction, we assume that each node p in the spanning tree knows the identity of all its direct children $\{q_0, \ldots, q_d\}$ in the tree. For each child q_i and every node identifier d, p can invoke a method `InSubtree`(d, q_i), which returns *true* if and only if d is in the sub-tree rooted at q_i. Such a service can be implemented easily in recent wireless protocols using IEEE 802.15.4 as a dedicated hierarchical addressing scheme based on a parent–child relationship has been designed. In other older protocol, this service can also be realistically implemented, e.g., by having every node, starting at the leaves, periodically notify its parent in the tree about all its known decedents. This may be either by reporting the entire list of decedents, or by passing the corresponding Bloom filter tables [12]. Thus, the local invocation of the method `InSubtree`(d, q_i) can be evaluated very efficiently and without any additional communication. Of course, when the network is very mobile, this service might return stale information. Thus, the above implementation is reasonable for an environment where mobility happens at

[4] Our simulations are performed using a simulator that simulates true wireless network behavior, including real power propagation models, distortions, etc. with the 802.11 (WiFi) MAC protocol.

walking speeds, but is not suitable for inter-vehicular networks.

4. The routing protocols

A common part in all our protocols is that they assume that nodes can efficiently discover the path that goes along the tree edges between every pair of nodes *s* and *d*. Specifically, the idea is that every node *s* wishing to send a message to *d* for the first time, when the route to *d* is still not known, propagates a `find-tree-path` message `m` up the tree until it reaches the common ancestor of both *s* and *d*. From that point on, `m` traverses down the tree until it reaches the destination *d*. Moreover, `m` records the list of nodes it has passed through in the tree. When *d* receives `m`, it replies with a `path-found` message, which is accompanied with the reverse route R_T that was found by `m`, and is propagated according to R_T. When this response message reaches *s*, then *s* stores R_T and uses it as an initial path for all communication with *d*. Given the function `InSubtree()` we assumed in Section 3.2, the ideas above can be easily accomplished as listed in Fig. 1.

Note that the cost of finding such a route along tree edges is relatively small. The number of messages sent is linear with the length of this path. Thus, throughout the rest of this section we assume the existence of an initial route like the one we described above. We concentrate on improving such routes if the communication between the corresponding nodes merits further investment in optimizing the route length.

4.1. The spanning tree (ST) protocol

The first protocol we explore, which we call spanning tree (or ST for short), always uses the tree route that is discovered by the algorithm in Fig. 1. In other words, ST simply invokes the *sendSpanningTree*(s, d, msg) method for each message. The advantage of this protocol is that once a tree is built, it does not waste any messages on path searching. Its only cost comes from the fact that messages are sent along sub-optimal routes.

To get a feel for the behavior of this simple protocol and to motivate the rest of this work, we have conducted the following simple experiment using the SWANS/JIST simulator [1]. We created random networks of 512 nodes in which one arbitrarily chosen node *s* sends a message to every other node, in some random order, and then *s* repeats this process iteratively for a given number of times. We then record the total accumulative number of transmissions from the beginning of the simulation until the end of each iteration. The results, which are an average of 10 runs each, are illustrated in Fig. 2 (in the case of ST, it includes the cost of the tree maintenance).

```
FindInitialPath(s,d)
    HandleTreePathSearch(s,d,{p})

HandleTreePathSearch(s,d,R_T)
    R_T := R_T + my_id;
    if my_id = d then
        send <path-found,R_T > to last node in R_T
    else if (InSubtree(d,q_i) = false) for all my children {q_i} then
        send <find-tree-path,s,d,R_T > to my parent in the tree
    else
        let q_i be my child for which InSubtree(d,q_i) = true;
        send <find-tree-path,s,d,R_T > to q_i
    endif

Upon receiving a <find-tree-path,s,d,R_T > message do
    HandleTreePathSearch(s,d,R_T)
enddo

Upon receiving a <path-found,R_T > message do
    if my_id is the first identifier in R_T then
        invoke PathFound(s,d,R_T)
    else
        let next := the node identifier before my_id in R_T;
        send <path-found,R_T > to next
    endif
enddo
```

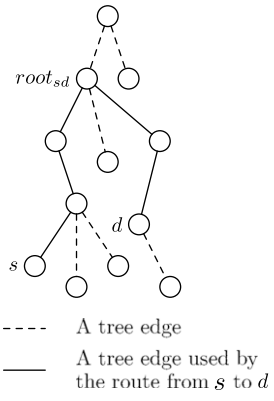


Fig. 1. Finding an initial path between *s* and *d*.

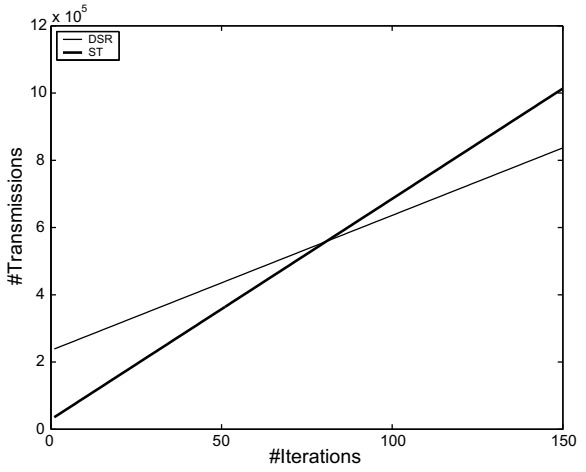


Fig. 2. Accumulative transmission costs of ST and DSR.

As can be seen, when the iteration number is lower than 80, ST generates fewer message. This indicates that when a pair of nodes do not communicate very frequently, at least before a major network configuration change, then ST is more effective than DSR. As the frequency of communication increases, DSR becomes better. The reason is that DSR initially pays a high price for the search flood that it performs in order to find the shortest path. However, once the shortest path has been found, it only pays a minimal cost of the shortest path for each additional application message. In contrast, ST does not spend any effort in trying to find a good route, and thus its initial cost is 0, but its per message cost is higher, since it routes messages over sub-optimal paths.

Of course, the results of the above test should be taken with a grain of salt. This is because, in practice, we do not know ahead of time which pairs of nodes will communicate frequently, we have chosen one specific communication pattern in this experiment, and we are ignoring here several other issues. However, it does suggest that there is a good potential for reducing the communication load by utilizing the spanning tree, which is what we explore below.

4.2. The Hybrid protocol (HYB)

When considering Fig. 2, ideally we would like to find a protocol that behaves like ST when nodes communicate rarely, and like DSR when the communication is frequent enough to justify the cost of a flood search. The simplest idea, which we call Hybrid (HYB), is to use ST initially, and then switch to DSR if the accumulative cost is high enough.

We can use here an analogy from the well studied *ski-rentals* problem [35]. In this problem, a person who is starting to take ski lessons has to decide whether to buy the ski equipment or to rent it. Renting once is cheaper than buying, but there is a per usage charge. Conversely, after purchasing the equipment, each additional usage is free. The problem is that this person does not know whether he will like skiing, and therefore does not know how often he will use the ski equipment. It turns out that the best on-line strategy for this problem is to rent the equipment until the accumulating cost of renting is as high as buying, and then to buy the equipment [35].

In HYB, we have an analogous problem, and therefore we would like to use the same solution. That is, ST is similar to the renting strategy, while DSR is similar to the purchasing strategy. Thus, in HYB we start by using ST. Once a given tree route is utilized often enough so that the accumulative cost for this path is similar to what DSR would have "paid", we switch to a DSR-like flood search, as illustrated in Fig. 3. Yet, to realize this scheme, we need a mechanism that can determine the cost of using ST compared to DSR. We describe such a mechanism below.

Formally, our goal is to detect whether on the nth usage of a tree route R_T we have accumulated a transmission overhead that is equal to the transmission cost of a flood search. To do this, we first need to estimate the overhead w accumulated by each usage of R_T (instead of an optimal path) and the cost c of a flood that will discover an alternative optimal route. Clearly, if after an iteration n, $nw \geqslant c$, then after the n iteration, the protocol has already accumulatively "paid" the same cost it would have "paid" had it used DSR right from the start. Hence, by the ski-rentals result, HYB switches from ST to a flood-based route discovery when $n \geqslant c/w$. We denote the length of a route R_T by $|R_T|$.

The overhead induced by the use of a tree route R_T rather than an optimal route R is $w = |R_T| - |R|$. As we do not know R ahead of time, we estimate $|R|$ as $|R_T|/(|\bar{R_T}|/|\bar{R}|) = \frac{|R_T|}{\rho}$ where $|\bar{R}|$ is the expected average optimal path length and $|\bar{R_T}|$ is the expected average tree length (so ρ is the estimated ratio between an average tree route and an average optimal route). Hence, the expected value for $w = |R_T| - |R| \approx |R_T| - \frac{|R_T|}{\rho} = |R_T| \cdot (1 - 1/\rho)$. ρ can be obtained empirically by experiments or set by some heuristic.

For example, assume that an average optimal route between two arbitrary nodes s and d is of length δ hops (for some δ). Hence, in this case, the average optimal route length between s, d and their common ancestor r is δ as well. As the tree route between s and d includes going from s to r and then from r ro d, and given the assumption about the tree properties in Section 3.2, the ratio ρ between the tree route and the optimal route can be estimated as $\frac{2\delta}{\delta} = 2$. Finally, since flooding the network implies a broadcast by each node we choose c to be N.

4.3. The Hybrid Tree-Bounded protocol (HYB_TB)

The next protocol we consider is the Hybrid Tree-Bounded protocol (HYB_TB). This protocol works like HYB, except that when we apply the flood search for finding the shortest path, the flood is restricted to the smallest common sub-tree that includes both the sender and the receiver. The rational is that if the network is well connected and uniformly distributed, then with high probability there is a path between any source s and any destination d that passes entirely through their common sub-tree, and is either a shortest path between them or very close to being the shortest. On the other hand, by restricting the floods to

local variables at each node

$R_{T_{sd}}$ // the spanning tree route from s to d
$R_{sd} \leftarrow \emptyset$ // the optimal route from node s to d (if known)
$count_{sd} \leftarrow 0$ // the number of times s has routed to d
ρ // the estimated ratio between an average tree route and an average optimal route

`sendHybrid`(s,d,msg)
if $R_{sd} \neq \emptyset$ **then**
`sendDsr`(s,d,msg)
else if $count_{sd} > \frac{N}{|R_{T_{sd}}|(1-\frac{1}{\rho})}$ **then** // this expression approximates c/w
// see explanation in the text
`sendDsr`(s,d,msg)
else
`sendSpanningTree`$(R_{T_{sd}},\ count_{sd})$
endif

Fig. 3. The Hybrid (HYB) scheme.

the common sub-tree, we can save many useless search messages.

As part of our basic assumptions on hybrid networks, each node x knows a route φ_x from itself to the spanning tree root. In particular, φ_x contains the roots of all sub-trees that include x. Considering now a flood search from a node s to a node d bounded to the sub-tree of their common ancestor r. Only nodes x for which $r \in \varphi_x$ will participate in the search. Furthermore, a node x will participate in the search only if its depth, $h(x)$, satisfies $h(r) \leqslant h(x) \leqslant h(s) + h(d) - h(r)$. We assumed in Section 3.2 that the spanning tree is built such that if nodes x and y are within transmission range of each other then $|h(x) - h(y)| \leqslant 1$. Thus, any monotonic route, i.e., a route that either only ascends through the tree or only descends through the tree, is a shortest route between s and d. Hence, any route that goes beyond these boundaries will be longer or equal to the tree route. This is because such a route traverses more tree levels than the length of the tree route.

In considering the pseudo-code of HYB_TB, as mentioned above, the discovery of optimal routes relies on a flood that is bounded to the sub-tree of the source and the destination. The details appear in Fig. 4.

4.4. The Hybrid-Iterative protocol (HYB_ITR)

Our next protocol is a refinement of HYB. That is, in HYB, we wait until the cost of using ST is the same as DSR, and only then we utilize a flood search for finding a shortest path. The idea in the Hybrid-Iterative protocol (HYB_ITR) is to apply the same principles of HYB, but in a gradual manner. That is, in each iteration n, if the expected cost of using ST n times is already as high as performing a limited flood search with $TTL = t$, for some t, and we have not issued a flood search for this path with $TTL = t$ before, then we issue a corresponding flood search (with $TTL = t$). A pseudo-code description of this protocol appears in Fig. 5.

As in HYB, the protocol needs to detect whether on the nth usage of a tree route R_T we have accumulated a transmission overhead that is equal to the transmission cost of a flood search bounded by $TTL = t$. As before, we estimate w to be $|R_T| \cdot (1 - \rho)$. A flood with $TTL = t$ implies a broadcast of all nodes within $t - 1$ hops from the source, or from a geometric perspective, all nodes of distance less than or equal to $(t - 1)\mathscr{R}$ from the source, where $\mathscr{R}$ is the transmission radius. To that end the cost of a flood with $TTL = t$ is denoted $c(t) \approx \frac{\pi[\mathscr{R}(t-1)]^2}{A^2}N$, where A is the dimension of the square that bounds the network.

4.5. The Hybrid Shortcut protocol (HYB_SC)

Our last protocol is a combination of ideas from HYB_TB and HYB_ITR. Specifically, in HYB_SC, we try to look for shortcuts inside the common sub-tree of s and d using limited floods. Each flood can only propagate inside the common sub-tree, and is limited by a TTL which is computed in the same way as in HYB_ITR. Moreover, rather than only searching for a path from s to d, here we are looking for any possible shortcut for the tree-based path between s and d. Moreover, based on the depth of the common sub-tree between s and d and the TTL value k, we decide in what parts of the tree we are likely to find such shortcuts, and further restrict the flood to propagate only in that part of the sub-tree.

Specifically, HYB_SC initiates the flood search similarly to HYB_TB. However, in HYB_SC it is not the source node that initiates the flood. Instead, the source node detecting that it is time to trigger a flood with $TTL = t$ piggybacks that information on the routed message and sends it along the tree route. When the message is processed by a tree route node x that precedes the sub-tree root by t hops, then x initiates a flood with $TTL = t$. Furthermore, the flood is bounded to the depth of $h(x) \pm \log(t)$. We expect tree shortcuts to mainly extend through similar tree levels. In other words, when taking a graphic representation of the tree, as in Fig. 6, the shortcuts are likely to have a horizontal orientation. We therefore force the floods to expand along similar tree depths, or horizontally in the graphical representation, in order to save useless search messages. Yet, as the TTL grows, we allow more "vertical freedom"

local variables at each node
$R_{T_{sd}}$ // the spanning tree route from s to d
$R_{sd} \leftarrow \emptyset$ // the optimal route from node s to d (if known)
$count_{sd} \leftarrow 0$ // the number of times s has routed to d
$root_{sd}$ // the common ancestor of s and d in the spanning tree
φ // the route from the local node to the spanning tree root
ρ // the estimated ratio between an average tree route and an average optimal route

`sendHybridTreeBounded`(s,d,msg)
if $R_{sd} \neq \emptyset$ **then**
`sendDsr`(s,d,msg)
else
if $count_{sd} > \frac{N}{|R_{T_{sd}}|(1-\frac{1}{\rho})}$ **then** // this expression approximates c/w (see Section 4.2)
`treeFloodRouteDiscovery`(s,d,$|R_{T_{sd}}|$)
endif
`sendSpanningTree`($R_{T_{sd}}$, $count_{sd}$)
endif

`treeFloodRouteDiscovery`(s,d,ttl) // start a tree bounded flood search
$minDepth = h(root_{sd})$
$maxDepth = h(s) + h(d) - h(root_{sd})$
`send`($treeFloodRouteRequest < s, d, ttl, root_{sd}, minDepth, maxDepth, \{s\} >$)

`handleTreeFloodRouteRequest`(s,d,ttl,$root$,$minDepth$,$maxDepth$,$recordRoute$)
if $root \in \varphi$ **and** $minDepth \leq my\_depth \leq maxDepth$ **then**
if $d = my\_id$ **then**
`send`($treeFloodRouteResponse < s, d, recordRoute \bigcup \{my\_id\}, reverse(recordRoute) >$)
else if $ttl > 0$ **then**
$recordRoute \leftarrow recordRoute \bigcup \{my\_id\}$
`send`($treeFloodRouteRequest < s, d, ttl - 1, r, minDepth, maxDepth, \{s\} >$)
endif
endif

`handleTreeFloodRouteResponse`(s,d,$routeResponse$,$sourceRoute$)
if $s = my\_id$ **then**
$R_{sd} \leftarrow routeResponse$
elseif $sourceRoute[0] == my\_id$ **then**
`send`($treeFloodRouteResponse < s, d, routeResponse, sourceRoute \setminus sourceRoute[0] >$)
endif

Fig. 4. The Hybrid Tree-Bounded (HYB_TB) scheme.

to increase the odds of finding shortcuts when seeking deeper in the tree. The idea here is that the longer the shortcut we are looking for, the higher the chances are that such a shortcut will span more levels in the tree. Fig. 6 shows which nodes will initiate floods on the second, fourth and ninth attempts to shortcut a tree route and the depth boundaries of those floods. The detailed HYB_SC protocol is listed in Fig. 7.

5. Simulations

We have compared all the above mentioned schemes to each other, as well as to DSR, and a variant of DSR, which we call DSR_TB, in which the search process of DSR is bounded to the minimal common sub-tree of the source and destination. All simulations were carried with the SWANS/JIST simulator [1]. In all simulations, nodes were placed randomly with uniform distribution in the simulation area and were static throughout the simulation; the transmission range was set to $R = 200$ m,[5] the bandwidth to 54 MB/s, and the simulation area to $A \times A$ m^2, where $A = [\pi(R^2)N]/[X \ln(N)](X = 3)$, so for $N = 512, A = 1854.15$, and for $N = 1024, A = 2487.6$. This calculation of the size is based on the results of [22] for ensuring that the network will be connected with very high probability. SWANS emulate the 802.11 MAC protocol where nodes use two-ray ground radio propagation model, which also accounts for distortions and collisions. In these experiments, we compared the overhead in terms of total number of message transmitted under different communication patterns. In

[5] To be precise, the radio frequency was set to 2.4 GHz, transmission strength at 15 dB m with antenna gain of 0 dB, radio reception sensitivity −80.0 dB m and radio reception threshold −71.1 dB m with the independent radio noise level, temperature 290.0 K, temperature noise factor of 10, ambient noise 0 mW, and signal to noise ratio threshold of 10.0. Also, both the RX–TX switching delay and the physical layer delay were set to 5 μs.

local variables at each node

$R_{T_{sd}}$ // the spanning tree route from s to d
$R_{sd} \leftarrow \emptyset$ // the optimal route from node s to d (if known)
$count_{sd} \leftarrow 0$ // the number of time s has routed to d
$TTL_{sd} \leftarrow 1$ // the TTL of the next flood to be initiated at s seeking a shortest route to d
ρ // the estimated ratio between an average tree route and an average optimal route

`sendHybridItrative`(s,d,msg)
if $R_{sd} \neq \emptyset$ **then**
`sendDsr`(s,d,msg)
else
if $count_{sd} > \frac{c(TTL_{sd})}{|R_{T_{sd}}| \cdot (1-\frac{1}{\rho})}$ **then** // recall that $c(t) = \frac{\pi[\mathcal{R}(t-1)]^2}{A^2}N$
`floodRouteDiscovery`(s,d,TTL_{sd})
$TTL_{sd} := 2 \cdot TTL_{sd}$
endif
`sendSpanningTree`($R_{T_{sd}}$, $count_{sd}$)
endif

Fig. 5. The Hybrid-Iterative (HYB_ITR) scheme.

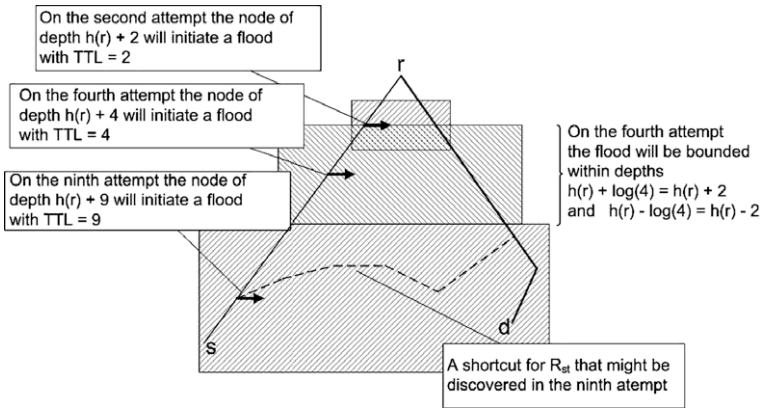


Fig. 6. Floods carried seeking for shortcuts in HYB_SC.

all tree-based scheme, this number includes the cost of constructing the tree. All the results that we present are an average of 10 runs.

A known extension to DSR, called *expanding ring* gradually increases the size of the flood search radius during the route discovery routine instead of immediately flooding the entire network. This yields a reduced number of transmissions at the expense of longer discovery delays. Hence, when comparing our protocols with DSR in terms of the number of transmissions, we apply the expanding ring extension and when comparing the discovery delay times we do not apply this extension. Thus, we always measure against the more efficient version of DSR with respect to the property being measured. See Section 5.3 and Appendix A for more details.

5.1. Single sender

In the first experiment we conducted, first, a node is randomly chosen from a set of 512 nodes. Then, the chosen node iteratively performs the following: in each iteration, it sends one message to each other node in the system. We have measured the total accumulated number of transmissions in the system at the end of each iteration. The results are plotted in Fig. 8 (and the zoomed in view on the first iterations in Fig. 9).

DSR is initially worse than the other schemes, but as the iteration number increases, DSR becomes relatively better. As discussed above, the reason is that DSR initially pays a high price to find the best route, but once this route is found, DSR pays an additional minimal additive cost, since from that point on it always utilizes the shortest path. The other schemes pays a small initial price, but their penalty on each iteration, for using a sub-optimal path, is higher than DSR. Also, the gradient of the schemes HYB and HYB_ITR eventually becomes the same as the one of DSR. This means that they eventually find the shortest path. In the case of DSR-TB, HYB_TB, and HYB_SC, the ultimate gradient is slightly worse, meaning that in a few cases (the graphs show the cumulative numbers), these schemes failed to find the optimal path. An interesting observation is the good performance of DSR_TB, which until iteration number

local variables at each node

$R_{T_{sd}}$ // the spanning tree route from s to d
$count_{sd} \leftarrow 0$ // the number of times s has routed to d
$root_{sd}$ // the common ancestor of s and d in the spanning tree
$level_{sd}$ // the tree level of the node that will initiate the next flood with TTL = $level_{sd}$
ρ // the estimated ratio between an average tree route and an average optimal route

`sendShortcuts`(s,d,msg)
$i \leftarrow$ [index of $root_{sd}$ in $R_{T_{sd}}$] - $level_{sd}$
$initiator = R_{T_{sd}}[i]$
$max\_depth = h(initiator) + \log(level_{sd})$
$min\_depth = h(initiator) - \log(level_{sd})$
$targets = \{\alpha \in R_{T_{sd}} \mid min\_depth \leq h(\alpha) \leq max\_depth\}$
if $count_{sd} > \frac{c(level_{sd})}{|R_{T_{sd}}| \cdot (1-\frac{1}{\rho})}$ **then** // recall that $c(t) = \frac{\pi[\mathcal{R}(t-1))]^2}{A^2} N$
$level_{sd} \leftarrow level_{sd} + 1$
`sourceRoute`($shortcutsMessage < R_{T_{sd}}, msg, initiator, max\_depth, min\_depth, level_{sd}, root_{sd}, targets >$)
else
`sourceRoute`($shortcutsMessage < R_{T_{sd}}, msg, \perp, \perp, \perp, \perp, \perp, \perp >$)
endif

`handleShortcutsMessage`($R, msg, initiator, max\_depth, min\_depth, ttl, root, targets$)
$q \leftarrow my\_id$
foreach $trg \in targets$
if $R_{qd} \neq \emptyset \wedge |R_{qd}| < |R_{T_{qd}}|$ **then** // if q knows a shortcut to d, it applies it
apply R_{qd}
endif
endfor
if $initiator = q$ **then**
`send`($treeFloodRouteRequest < q, targets, ttl, root, minDepth, maxDepth, \{q\} >$)
// see Fig. 4
endif

Fig. 7. The Hybrid Shortcuts scheme (HYB_SC).

265 is better than all the other schemes. Notice that the communication pattern used in this experiment is helpful for DSR-like protocols. The reason for this is that DSR remembers paths; if a sender s knows the path to some node d_1, and d_2 is an intermediate node on this path, then whenever s wishes to send a message to d_2, then s uses the corresponding prefix from the path it knows to d_1. In this test, we have a single sender sending messages sequentially to all other nodes. Hence, many of the paths are already known at the time of sending, since they are prefixes of paths that were discovered previously.

Fig. 8 represents a specific communication pattern. Still, it gives a feel for which protocol is preferable if we know the exact number of times a route will be used before it is broken.

5.2. All-to-all communication

Uniform distribution: In the second experiment, a subset of the nodes (participants) is picked at random. From this subset of nodes, we repeatedly randomly pick a pair of nodes using uniform distribution, and then have one of the nodes send a message to the other node. For a given number of participants P, we repeat this procedure P^3 times, and measure the total number of messages. Notice that as there are P^2 pairs of nodes, by choosing among them P^3 times, each pair has a very high probability of being selected at least once. We conducted this test both when the total number of nodes is 512 and the number of participants is 128 and when the number of nodes is 1024 and the number of participants is 80 (Fig. 10).

Heavy-tailed distribution: Our third experiment is similar to the second, except that the pair of nodes that communicate each time is chosen using a heavy-tailed Zipf-like distribution. Hence, in this case, a few nodes communicate very often, while the rest communicate rarely. The results are reported in Fig. 11.

Normal distribution: Our last experiment is the same as the previous two, but using normal distribution. The results are listed in Fig. 12.

Discussion: By looking at the graphs, we can observe several phenomena. First, the performance of the various schemes is highly affected by the communication pattern and the ratio between the overall network and the number of participating nodes.

For example, DSR behaves better when the percentage of participants is higher. This can be explained by a combination of several optimizations in DSR: First, the reuse of paths, as discussed before. As the number of participants becomes larger, it increases the chances of being able to re-

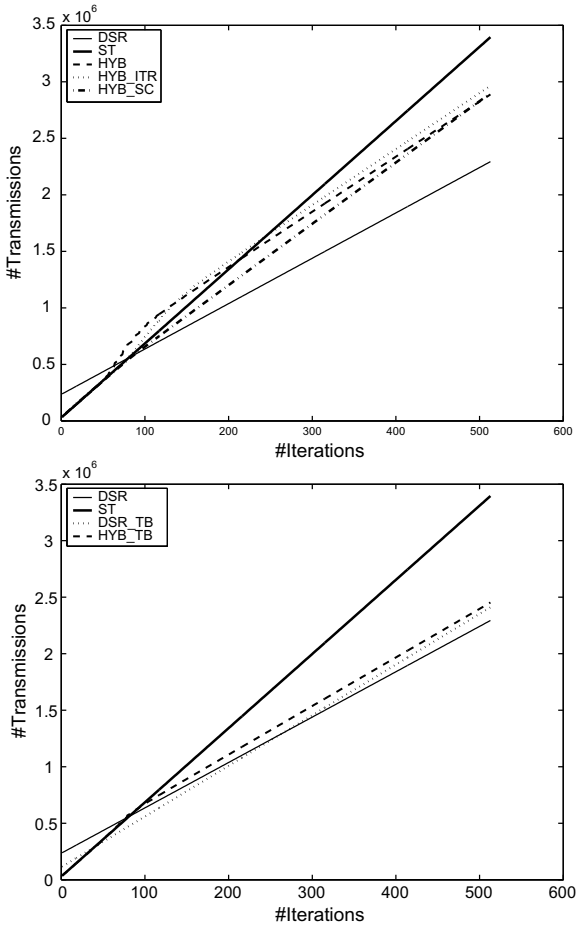


Fig. 8. Accumulative transmission cost in the one sender experiment.

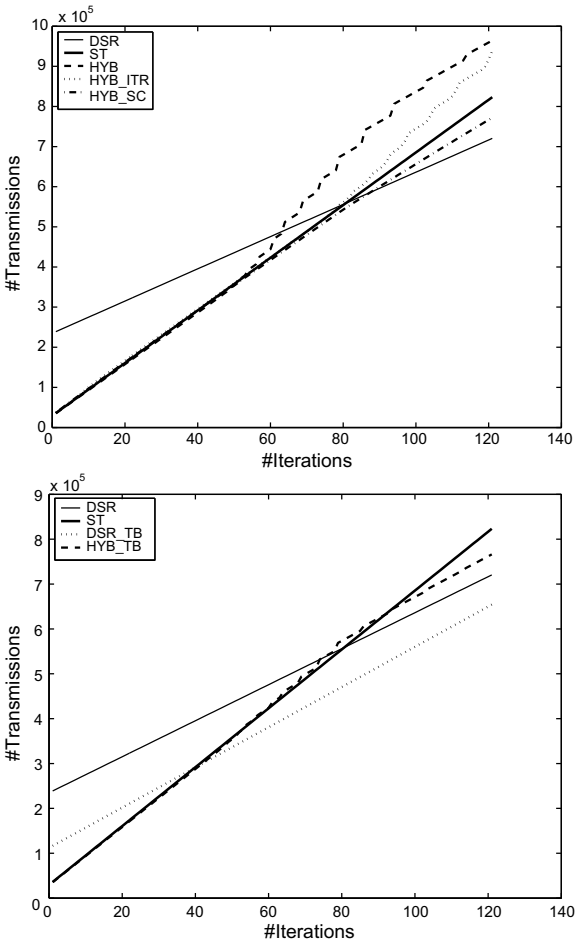


Fig. 9. One sender experiment – zooming in on the first iterations.

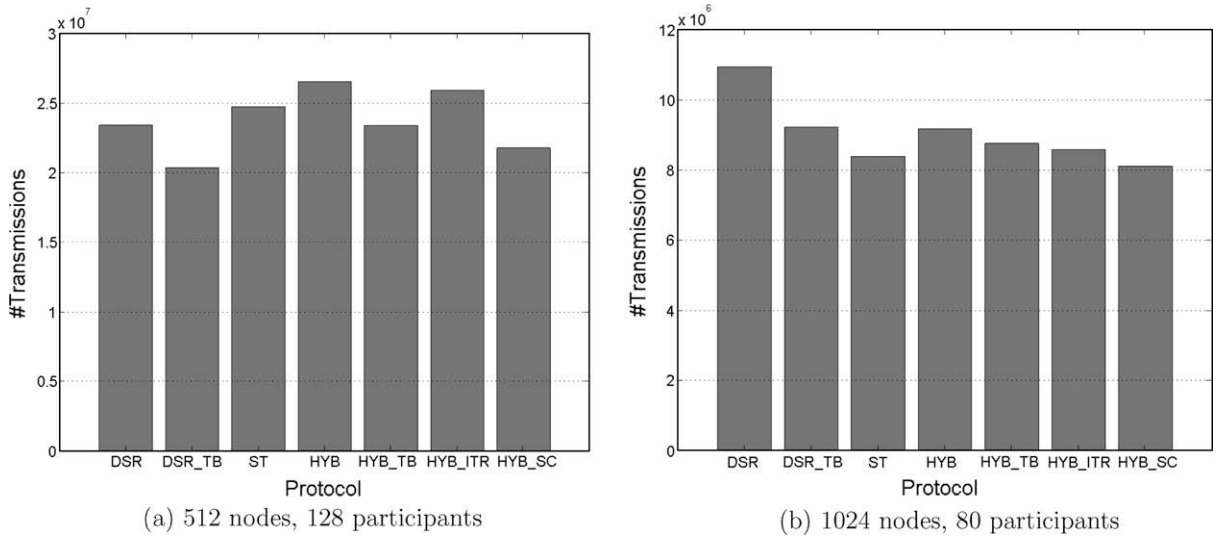
use paths. In particular, when a search request for a destination d reaches a node e that already knows a path to d, then e replies with this path, which truncates the flood, especially since DSR also uses an expanding ring search. Additionally, when comparing DSR to the other protocols, DSR behaves relatively better with uniform distribution. This is because in heavy-tailed distributions, a large number of process pairs communicate rarely. For these nodes, DSR pays the heavy cost of a flood search without the reward of repeated usage.

Second, when comparing the simple ST protocol to the others, ST behaves relatively better when the proportional number of participants is small, except for the case of normal distribution. In fact, in the uniform distribution with 80 participates out of 1024, ST was the second best protocol. In this settings, paths are rarely used, and thus the gain of using the shortest path does not make up for the cost of the flood searches. In the heavy-tailed distribution ST suffers from the fact that some paths are used very frequently, so the accumulated cost that ST pays on these paths is relatively high.

The HYB scheme works very well in the heavy-tailed distribution, given its adaptive nature. Yet, in uniform distribution, it suffers from the fact that most paths are used a large enough number of times to merit the flood search right from the start. For this reason, in the uniform distribution, HYB_ITR is better, as it starts searching for optimal paths sooner than HYB. Conversely, HYB_ITR is slightly worse than HYB in the heavy-tailed distribution because it often initiates (limited) floods when the utilization of the path is not sufficient to justify it.

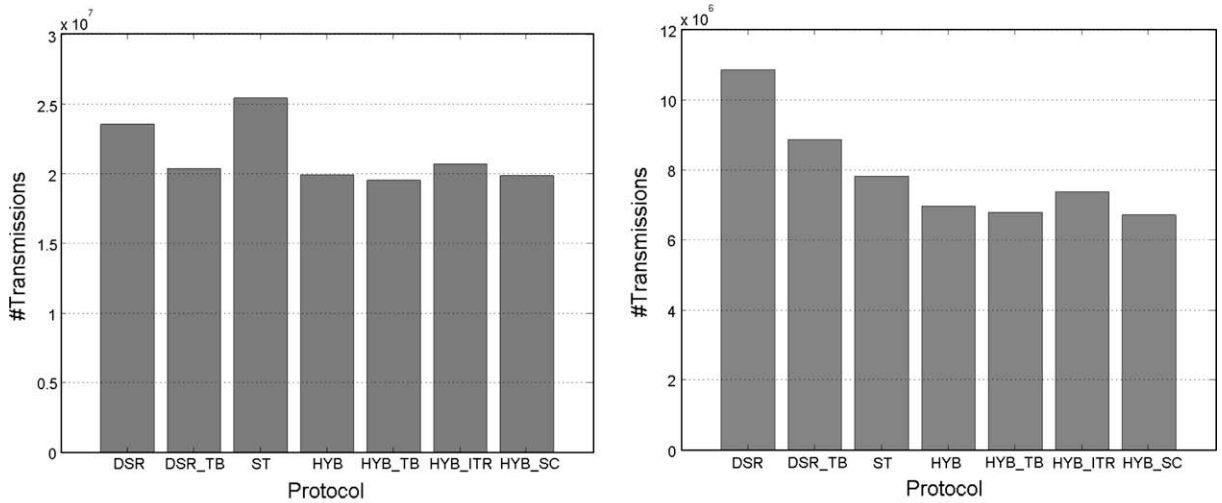
The DSR_TB protocol performed much better than the DSR protocol in most cases. As we were using uniformly random placement of the nodes, a path that is either the shortest or very close to the shortest is very likely to be bounded inside the common sub-tree of every sender and every receiver. Hence, by bounding the search this area, DSR_TB can always find a short path with a greatly reduced overhead. Interestingly, in the case of HYB_TB, the benefits compared to HYB were much smaller. This is because HYB (and also HYB_TB) invokes a flood search in fewer cases than DSR.

The results of the normal distribution are overall similar to the uniform distribution. Yet, the optimized schemes (HYB, HYB_TB, HYB_ITR, and HYB_SC) perform a little better with the normal distribution compared to their performance with the uniform distribution. This is expected,



(a) 512 nodes, 128 participants (b) 1024 nodes, 80 participants

Fig. 10. Transmission costs with a uniform distribution.



(a) 512 nodes, 128 participants (b) 1024 nodes, 80 participants

Fig. 11. Transmission costs with a heavy-tailed distribution.

since in this experiment, most routes are utilized enough to justify floods for finding an optimal path. Yet, in the normal distribution some routes are not, which is where the optimized schemes benefit.

Finally, HYB_SC came out as the most promising scheme overall. In almost all of the scenarios it was the best, and in one case it came out second best by a negligible margin. This scheme adapts best its costs to the communication pattern, and therefore behaves better than the others.

5.3. Route discovery delays

Aside from evaluating our protocols in terms of number of transmissions, we also measured the route discovery delay time. As all protocols are based on a tree route known a priori, we measured, by simulation, the delay time of the route discovery procedure shown in Fig. 1 and compared it with the flood-based route discovery delay time of DSR. We found that the when DSR is invoked without the expanding ring optimization, that the tree-based route discovery is on average about 17 times faster than DSR. When the expanding ring optimization of DSR is utilized, than this ratio increases to 70.

As for the hybrid schemes, the first route is found with the delay time of ST. Further improvements occur in the background based on the path utilization frequency and do not delay the communication. Furthermore, a single improvement trial is clearly bounded by the delay of DSR, and can even be less if applied gradually.

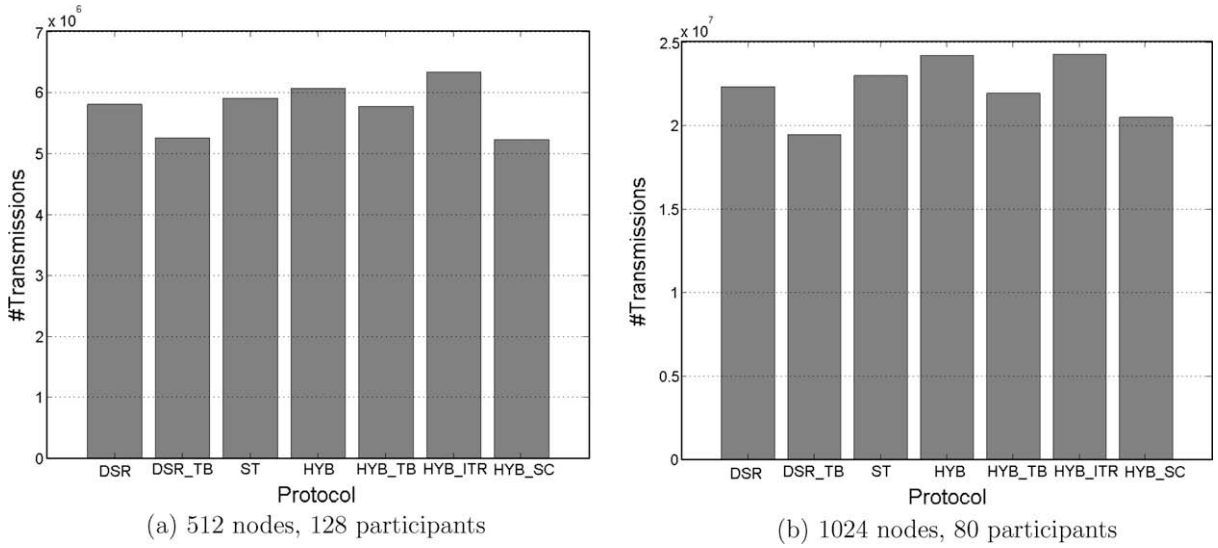


Fig. 12. Transmission costs with a normal distribution.

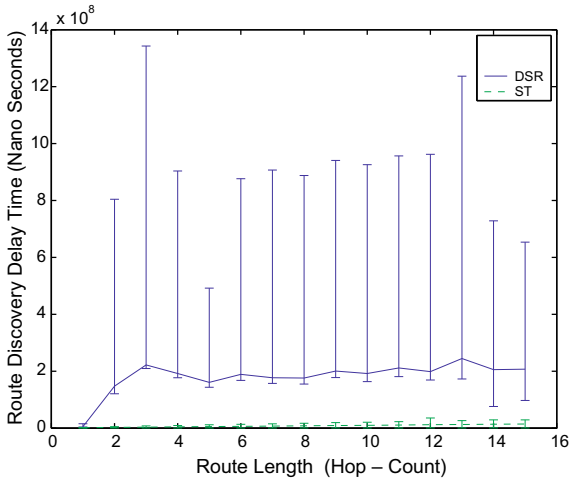


Fig. 13. Route discovery delay of ST vs. DSR (without expanding ring).

Notice that flood-based route discovery procedures propagate a route request through the network using broadcasts. In order to avoid the known *Broadcast storm* effect, a broadcast jitter randomly delays the retransmission of the query and therefore the procedure is delayed. Since the tree-based route discovery procedure uses point-to-point communication, no such jitter mechanism is necessary, so no random artificial delay is required. Consequently, the overall delay is shorter and more predictable as can be seen in Fig. 13. In particular, the large error bars for DSR are the accumulative result of the random jitter at each hop.

6. Competitive analysis

In all the protocols described in Section 4 above, the amount of effort to find a path between two nodes depends on its popularity. Moreover, the route length used for such communication monotonically improves with its utilization. On the other hand, a hypothetical off-line protocol that knows a priori how many times a certain route will be used could predetermine whether to invest in discovering an optimal route, or to be satisfied with a sub-optimal route absorbing the per usage transmission overhead. By definition, such a protocol does not exist. Yet, it serves as a baseline to which we can compare our algorithms. Specifically, we perform a competitive analysis of some of the on-line hybrid protocols we proposed against such a hypothetical off-line protocol. The analysis is carried under the assumption that we can estimate the difference between the optimal route and tree route accurately.

We start the analysis by computing the overall routing cost that the optimal off-line protocol needs to pay. For this, consider the task of discovering a route R from node s to node d and suppose that this route is used n times. We denote the cost of discovering an optimal route through flood search by F. Yet, once an optimal route is discovered, its per usage cost (for the purpose of the competitive analysis) is zero. However, if the route is a tree route, its discovery cost is zero and its per usage cost is denoted by w.

As common in competitive analysis, we investigate the performance of the on-line protocols when operating against an adversary. The adversary is given the power to chose n and a routing strategy for the off-line protocol. Notice that there is no sense for an adversary to apply a strategy that routes both on a tree route and an optimal route; an off-line strategy that eventually floods the network for an optimal route might as well do it right on the start. Thus, we can let the adversary protocol A, run two off-line instances, A_{st}, which routes only along the tree route with an accumulative cost nw and A_{dsr}, which runs only on the optimal route with an accumulative cost F. For each instance, n is chosen to maximize the competitive ratio between the off-line instance's accumulative cost and the on-line protocol's accumulative cost.

We now examine the competitive ratio of HYB. Recall that HYB uses the tree route for $k = \frac{F}{w}$ times and then pays F to switch to optimal path routing. Let us now consider the following cases. Suppose the adversary uses A_{st}. In this case, if $n < k$, then both HYB and the adversary use the same mechanism, and thus the competitive ratio is 1. However, if $n \geqslant k$, then HYB would pay $kw = F$ for its usage of tree routes and then an additional F for finding the optimal path, or a total of $2F$. The best strategy for the adversary is then to stop the run with $n = k$ after which A_{st} accumulated a cost of $wk = F$. In other words, in this scenario, the competitive ratio is 2.

Alternatively, using similar arguments, if the adversary uses A_{dsr}, the competitive ratio is $\frac{nw}{F} < 1$ for $n < k$ and 2 for $n \geqslant k$. Thus, in the worst case, the competitive ratio between the on-line protocol HYB and the off-line adversary protocol A is 2. This fits well with the result of the on-line ski rental problem [35] which also shows that setting $k = \frac{F}{w}$ is optimal.

We now turn our attention to HYB_ITR. We assume that the optimal path discovery mechanism performs limited scope flooding of the network with exponentially expanding *TTLs* [32]. Thus, the cost of a flood search by HYB_ITR can be written as $F = f_0 + f_1 + \cdots + f_q$ where f_i is the cost of a flood with $TTL = 2^i$. The strategy of HYB_ITR is to use the tree route while carrying the ith flood whenever the accumulated cost reaches f_i for some i and when the optimal route is found on the qth flood it starts routing along it. As we already mentioned above, the accumulated cost of A_{dsr} is constant and equals F. Thus, the worst ratio between HYB_ITR and A_{dsr} is achieved when the route is used a sufficient number of times so that HYB_ITR executes all the q iterations necessary to find the optimal route. The ratio obtained is then $\frac{\sum_{0 \leqslant i \leqslant q} f_i + f_q}{F} = \frac{F + f_q}{F}$.

On the other hand, A_{st} always uses the tree route. HYB_ITR also starts by using the tree route, so until it finds the optimal route, its accumulative cost is the cost of A_{st} plus the accumulated cost of the limited floods it performs. Hence, the worst case competitive ratio between HYB_ITR and A_{st} is also achieved when the route is used a sufficient number of times so HYB_ITR executes all the q floods necessary to find the optimal route. Recall that the behavior of HYB_ITR is such that it decides to issue the qth limited flood when the accumulated cost of using the tree routes is the same as the cost of such a flood. In other words, the accumulated cost of HYB_ITR for using tree route, and therefore also of A_{st}, at the point when the qth limited flood is issued is f_q. Consequently, in this case, the competitive ratio obtained is $\frac{\sum_{0 \leqslant i \leqslant q} f_i + f_q}{f_q} = \frac{F + f_q}{f_q}$.

Notice that $\frac{F}{f_q}$ is actually the ratio between the number of transmissions used to carry all floods $f_0, f_1, \ldots, f_q$ and the number of transmissions used to carry the last flood f_q. The cost of each flood is linear with the number of nodes participating in that flood. Recall that we assume that nodes are placed uniformly in the network and the *TTL* of the ith flood is 2^i. Moreover, the cost of a flood is linear with the number of nodes participating in this flood, since in a flood each participating node broadcasts the message once. Hence, the cost of the ith flood is linearly proportional to the area of the network covered by the flood. By ignoring edge effects, the area of a flood with $TTL = 2^i$ and a transmission radius of ρ can be approximated by $\pi(2^i\rho)^2$. Hence, $f_i \approx f_0 2^{i^2} = f_0 4^i$. Thus $\frac{F}{f_q} = \frac{\sum_{0 \leqslant i \leqslant q} f_i}{f_q} = \frac{\sum_{0 \leqslant i \leqslant q} f_0 4^i}{f_0 4^q} = \frac{\sum_{0 \leqslant i \leqslant q} 4^i}{4^q} = \frac{4^{q+1}-1}{3 \cdot 4^q} \approx \frac{4}{3}$. Based on this analysis, we can conclude that the worst case ratio of HYB_ITR and A_{dsr} is $\frac{F+f_q}{F} = 1 + \frac{3}{4}$ and the worst case ratio of HYB_ITR and A_{st} is $\frac{F+f_q}{f_q} = 1 + \frac{4}{3} = 2 + \frac{1}{3}$. Thus, HYB_ITR achieves a competitive ratio of $2 + \frac{1}{3}$.

7. Discussion and conclusions

In this paper, we have presented several schemes whose goal is to reduce the overhead of finding a path between two nodes in a hybrid network. This is by utilizing a spanning tree from the access point to all other nodes in it transitive transmission range. Generally speaking, the performance of the various methods depends on the communication pattern, and in particular on the discrepancy between the popularity of the various paths. Yet, one scheme, HYB_SC, performed overall better than the others.

7.1. Beyond hybrid networks

The motivation for our work comes from hybrid networks, in which we assumed a spanning tree from the access point to mobile nodes in its transitive transmission range will be maintained in any case. Clearly, the techniques can be applied to any ad-hoc network by picking an arbitrary node to be the root and building a spanning tree from this node to all others. In such cases, the cost of building and maintaining the spanning tree has to be accounted for in the formal analysis (recall that the simulations include in any case the cost of building a tree). Notice, however, the communication cost of maintaining a tree, at least in networks in which the mobility is mainly due to walking and running, is very similar to the cost of route maintenance in protocols like AODV. In both cases, the maintenance cost is mainly periodic sending of heartbeat messages between direct neighbors, and maybe piggybacking on these messages additional control information. Estimating the impact of such background periodic traffic in a formal competitive analysis ratio model is difficult.

7.2. Mobility

Due to mobility of nodes, the topology of the ad-hoc part of the network might change, causing routes between nodes to break. The protocols described previously can be naturally extended to support nodes mobility by overcoming routes breakups. First, the spanning tree must be maintained in the face of topology changes. This can be achieved with a relatively low cost by having nodes exchanging heartbeats with their parents and having the tree root periodically propagating a *beam* message down all the tree branches letting orphan nodes obtain a new parent. Second, broken routes should be discovered and replaced. This can be done, e.g., with common techniques that are also in use in DSR [32]. Specifically, a hop by hop acknowledge-

ment is needed so routing on a broken route can be discovered and nodes can be informed. The route can either be repaired locally or replaced with a rediscovered route. Since the tree is continuously maintained, no complete rediscovery is needed in this case.

Moreover, when nodes apply the incremental strategies described at Section 4, then in fact routing between rapidly moving nodes is always done over the spanning tree, refraining from vainly flooding the network. On the other hand routing between stationary nodes is done on shortest paths. In particular, in HYB_SC, the length of the route being utilized between any two nodes is inversely proportional to these nodes mobility rate. HYB_SC also poses a redundancy in the sense that while one shortcut may break, a different shortcut might still be valid so the route quality does not degrade drastically when a specific shortcut breaks.

Acknowledgements

We would like to thank Sidi-Mohammed Senouci and Bing Han for interesting comments and discussions. We also wish to thank the anonymous reviewers for their insightful comments that helped improve the presentation of this paper.

Appendix A. On the impact of expanding ring in DSR's message overhead

As mentioned in Section 5, we have applied the expanding ring extension to the DSR implantation used in the simulations (of transmissions overhead). In that implementation, the DSR route discovery routine starts its flood search with a radius of one hop and multiply the flooding radius by two until the target is found. We have tested the performance of DSR with and without this extension, and the results of these simulations are reported in Fig. A.1. In each experiment, a subset A of P nodes out of a total of N nodes invoked DSR in order to find all routes between all pairs of nodes in A (a total of $P(P-1)$ routes).

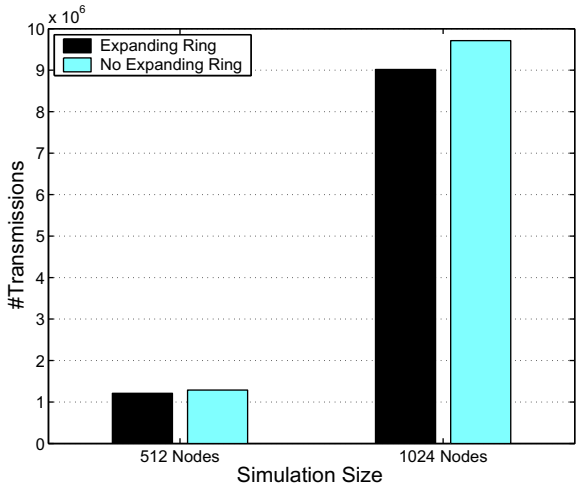


Fig. A.1. Impact of expanding ring on DSR.

No additional traffic was generated during the test. We conducted the experiment for $N = 512$, $P = 32$ and $N = 1024$, $P = 64$, taking the average of 10 simulation runs for each case. As can be seen, DSR generates fewer messages when the expanding ring extension is enabled than without it. Specifically, the difference was 6.5% in the 512 nodes network and 7.7% in the 1024 nodes network (in favor of DSR with expanding ring).

References

[1] Swans/jist. <http://jist.ece.cornell.edu/>.
[2] ZigBee alliance document 053474r17: Zigbee specification, January 2008.
[3] IEEE 802.15.4 standard, September 2006.
[4] A. Agarwal, P.R. Kumar, Capacity bounds on ad hoc and hybrid wireless networks, ACM SIGCOMM Computer Communications Review 34 (3) (2004) 71–81.
[5] K.M. Alzoubi, P.-J. Wan, O. Frieder, Message-optimal connected dominating sets in mobile ad hoc networks, in: Proc. of the Third ACM International Symposium on Mobile Ad Hoc Networking & Computing, 2002, pp. 157–164.
[6] A. Amis, R. Prakash, T. Vuong, D.T. Huynh, MaxMin D-cluster formation in wireless ad hoc networks, in: Proc. of IEEE Conference on Computer Communications (INFOCOM), March 1999.
[7] A.D. Amis, R. Prakash, Load-balancing clusters in wireless ad hoc networks, in: Proceedings of the Third IEEE Symposium on Application-Specific Systems and Software Engineering Technology (ASSET), 2000, p. 25.
[8] O. Arpacioglu, Z.J. Haas, On the scalability and capacity of planar wireless networks with omnidirectional antennas, Wireless Communication amd Mobile Computing 4 (3) (2004) 263–279.
[9] L. Badia, M. Miozzo, M. Rossi, M. Zorzi, Routing schemes in heterogeneous wireless networks based on access advertisement and backward utilities for QoS support, IEEE Communication Magazine 45 (2) (2007) 67–73.
[10] S. Banerjee, S. Khuller, A clustering scheme for hierarchical control in multi-hop wireless networks, in: Proc. of IEEE Conference on Computer Communications (INFOCOM), April 2001.
[11] P. Basu, N. Khan, T.D.C. Little, A mobility based metric for clustering in mobile ad hoc networks, in: Proc. International Workshop on Wireless Networks and Mobile Computing (WNMC), April 2001, pp. 16–19.
[12] B.H. Bloom, Space/time tradeoffs in hash coding with allowable errors, Communications of the ACM 13 (7) (1970) 422–426.
[13] J. Broch, D. Maltz, D. Johnson, Supporting hierarchy and heterogeneous interfaces in multi-hop wireless ad hoc networks, in: Proc. of the Fourth International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN), June 1999, pp. 370–375.
[14] I. Chlamtac, S. Kutten, Tree-based broadcasting in multihop radio networks, IEEE Transactions of Computers 36 (10) (1987) 1209–1223.
[15] T. Clause, P. Jacquet, A. Laouti, Optimized link state routing protocol, in: Proc. IEEE International Multi Topic Conference (INMIC), December 2001.
[16] F. Cuomo, S. Della Luna, U. Monaco, F. Melodia, Routing in ZigBee: benefits from exploiting the IEEE 802.15.4 association tree, in: Proc. of IEEE International Conference on Communications (ICC'07), 2007, pp. 3271–3276.
[17] B. Das, V. Bharghavan, Routing in ad-hoc networks using minimum connected dominating sets, in: Proc. of the 32nd IEEE International Conference on Communications (ICC), 1997, pp. 376–380.
[18] S.N. Diggavi, M. Grossglauser, D. Tse, Even one-dimensional mobility increases the capacity of wireless ad-hoc networks, in: Proc. of IEEE ISIT, Lausanne, Switzerland, June 2002.
[19] Y. Fernandess, D. Malkhi, K-clustering in wireless ad hoc networks, in: Proc. of the Second ACM International Workshop on Principles of Mobile Computing, 2002, pp. 31–37.
[20] R. Friedman, M. Gradinariu, G. Simon, Locating cache proxies in MANETs, in: Proc. Fifth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), May 2004, pp. 175–186.
[21] M. Grossglauser, D. Tse, Mobility increases the capacity of ad-hoc wireless networks, IEEE/ACM Transactions on Networking 10 (4) (2002) 477–486.
[22] P. Gupta, P.R. Kumar, Critical Power for Asymptotic Connectivity in Wireless Networks, Stochastic Analysis, Control, Optimization and

Applications: A Volume in Honor of W.H. Fleming, 1998, pp. 547–566.
[23] P. Gupta, P.R. Kumar, The capacity of wireless networks, IEEE Transactions on Information Theory 46 (2) (2000) 388–404.
[24] P. Gupta, P.R. Kumar, Internets in the sky: the capacity of 3 dimensional wireless networks, Communications in Information and Systems 1 (1) (2001) 33–50.
[25] J.Y. Ha, H.S. Park, S. Choi, W.H. Kwon, Ehrp: enhanced hierarchical routing protocol for ZigBee mesh networks, IEEE Communications Letter (2007) 1028–1030.
[26] Z. Haas, A new routing protocol for the reconfigurable wireless networks, in: Proc. IEEE Int. Conf. on Universal Personal Communications (ICUP), October 1997.
[27] C.-Y. Hsu, J.-L.C. Wu, S.-T. Wang, Capacity upgrading in mobile ad hoc access networks (MAHANs) using CSMA/CAPA, in: Proc. of the International Workshop on Mobility Management and Wireless Access (MobiWac), ACM Press, 2006, pp. 28–34.
[28] C.-Y. Hsu, J.-L.C. Wu, S.-T. Wang, Support of efficient route discovery in mobile ad hoc access networks, IEICE Transactions on Communications E89 (B(4)) (2006) 1252–1262.
[29] S. Hussain, O. Islam, An energy efficient spanning tree based multi-hop routing in wireless sensor networks, in: Proc. of IEEE Conference on Wireless Communications and Networking WCNC, 2007, pp. 4383–4388.
[30] I. Ioannidis, B. Carbunar, Scalable routing in hybrid cellular and ad-hoc networks, in: First IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS), October 2004, Poster.
[31] M. Jiang, J. Li, Y.C. Tay, Cluster Based Routing Protocol (CBRP), IE TF Internet Draft draft-ietf-manet-cbrp-spec-01, July 1999.
[32] D.B. Johnson, D.A. Maltz, Dynamic source routing in ad hoc wireless networks, in: Imielinski, Korth (Eds.), Mobile Computing, vol. 353, Kluwer Academic Publishers, 1996.
[33] U. Jonsson, F. Alriksson, T. Larsson, P. Johansson, G.Q. Maguire, MIPMANET – mobile IP for mobile ad hoc networks, in: Proc. of the First Workshop on Mobile Ad hoc Network and Computing (MobiHoc'00), August 2000, pp. 75–85.
[34] B. Karp, Geographic Routing for Wireless Networks, Ph.D. thesis, Harvard University, 2000.
[35] R. Karp, On-line algorithms versus off-line algorithms: how much is it worth to know the future? in: Proc. of the 12th IFIP World Computer Congress, 1992, pp. 416–429.
[36] Y. Ko, N.H. Vaidya, Geocasting in mobile ad hoc networks: location-based multicast algorithms, in: Proc. Second IEEE Workshop on Mobile Computer Systems and Applications, 1999, p. 101.
[37] P. Krishna, N. Vaidya, M. Chatterjee, D. Pradhan, A cluster-based approach for routing in dynamic networks, in: ACM SIGCOMM Computer Communication Review, April 1997, pp. 49–65.
[38] L. Lao, J.-H. Cui, M. Gerla, Tackling group-to-tree matching in large scale group communications, Computer Networks 51 (11) (2007) 3069–3089.
[39] H. Lei, C.E. Perkins, Ad hoc networking with mobile IP, in: Proc. of the Second European Personal Mobile Communications Conference, October 1997, pp. 197–202.
[40] J. Li, J. Jannotti, D.S.J. De Couto, D.R. Karger, R. Morris, A scalable location service for geographic ad hoc routing, in: Proc. Sixth Annual International Conference on Mobile Computing and Networking (MobiCom), 2000, pp. 120–130.
[41] W. Peng, Z. Li, F. Haddix, A practical spanning tree based MANET routing algorithm, in: Proc. of 16th IEEE International Conference on Computer Communications and Networks (ICCCN), 2005, pp. 19–24.
[42] C. Perkins, Ad Hoc on Demand Distance Vector (AODV) Routing, Internet Draft, draft-ietf-manet-aodv-00.txt, <citeseer.nj.nec.com/article/perkins99ad.html>, 1997.
[43] A. Rao, C. Papadimitriou, S. Shenker, I. Stoica, Geographic routing without location information, in: Proc. Ninth Annual International Conference on Mobile Computing and Networking (MobiCom), 2003, pp. 96–108.
[44] J.-H. Song, V.W.S. Wong, V.C.M. Leung, Efficient on-demand routing for mobile ad hoc wireless access networks, IEEE Journal on Selected Areas in Communications 22 (7) (2004).
[45] S. Srivastava, R.K. Ghosh, Cluster based routing using a *k*-tree core backbone for mobile ad hoc networks, in: Proc. of the Sixth International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM), 2002, pp. 14–23.
[46] Y. Sun, E.M. Belding-Royer, Application-oriented routing in hybrid wireless networks, in: Proc. Next Generation Internet Symposium, May 2003.
[47] Y. Sun, E.M. Belding-Royer, C.E. Perkins, Internet connectivity for ad hoc mobile networks, International Journal of Wireless Information Networks 9 (2) (2002). special issue on Mobile Ad hoc Networks.
[48] S. Toumpis, Capacity bounds for three classes of wireless networks: asymmetric, cluster, and hybrid, in: Proc. Fifth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), May 2004, pp. 133–144.
[49] J. Wu, M. Gao, I. Stojmenovic, On calculating power-aware connected dominating sets for efficient routing in ad hoc wireless networks, in: Proc. of the 30th International Conference on Parallel Processing (ICPP), 2001, pp. 346–353.
[50] J. Wu, H. Li, On calculating connected dominating sets for efficient routing in ad hoc wireless networks, in: Proc. of the Third Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communication (DialM), 1999, pp. 7–14.

Roy Friedman is an associate professor in the department of Computer Science at the Technion. He has published more than 70 technical papers on distributed systems, group communication, fault-tolerance, high availability, cluster computing, client/server middleware, and wireless mobile ad-hoc networks in major international journals and conferences. He also holds two patents. He also regularly participates in the technical program committees of leading international conferences. Formerly, He was an academic specialist at IRISA/INRIA (France) and a researcher at Cornell University (USA). He is one of the two technical founders of PolyServe Inc. (acquired by HP) and holds a Ph.D. and a B.Sc. from the Technion.

Ari Shotland graduated the Technion (Israel Institute of Technology) cum laude and received a B.Sc., in Computer Science. Currently, he is completing his M.Sc., at the Technion, conducting research in the area of wireless ad-hoc networks. During bachelor studies, He worked at IBM Haifa Research labs and currently he is working at Google Haifa.

Gwendal Simon received his Ph.D., degree in Computer Science in December 2004 after three years at both France Telecom R&D and IRISA. During his Ph.D., he most notably conceived and developed the Solipsis system – a distributed shared virtual world – which received a worldwide attention culminating at Codecon 2004. He then worked as a researcher at France Telecom R&D for almost two years where he focuses on spontaneous networks and peer-to-peer systems. He also implemented a strong collaboration with INRIA to form a noteworthy national research team in this area. Since September 2006, he is Associate Professor at TELECOM Bretagne, a graduate engineering school within the Institut TELECOM.