High Throughput Routing in Hybrid Cellular and Ad-Hoc Networks

Ioannis Ioannidis, Bogdan Carbunar, Cristina Nita-Rotaru
Purdue University
West Lafayette, IN, 47907
{ioannis,carbunar,crisn}@cs.purdue.edu

Abstract

Hybrid networks are a promising architecture that builds ad hoc, wireless networks around the existing cellular telephony infrastructure and supporting massive deployment for ad hoc networking. In this paper we present a routing protocol, DST, for hybrid networks that maintains a close to optimal spanning tree of the network by using distributed topology trees. DST is fully dynamic and generates only $O(\log n)$ *messages per update operation. We demonstrate experimentally that the performance of DST scales well with the network size and activity, making it ideal for the metropolitan environment hybrid networks are expected to operate in.*

1 Introduction

The past decade has witnessed rapid developments in wireless communications, from wireless cellular telephony to wireless LANs and PANs. Wireless network cards have become affordable and wireless connections are fast enough for users to abandon more traditional networking possibilities, as long as there is a nearby access point. The only factor against an explosion of wireless computer networking is the necessity for an expensive infrastructure that can provide extensive and reliable coverage with sufficient bandwidth.

Current 3G implementations, e.g. of W-CDMA in Japan, provide downlink rates of up to 380Kbps, but their promise for the close future is 2.0Mbps (2.4Mbps for cdma2000 1xEV-DO). The coverage area of a cellular base station is on the order of a few kilometers. Nevertheless, due to path loss via distance attenuation, the achievable rate significantly drops as the client moves away from the cellular base station. Furthermore, the transmission rate can be extremely erratic, making the network unreliable. Upgrading cellular base stations can solve these problems, although it is doubtful that providers will be willing to make such a massive investment.

While Wi-Fi hotspots are already being used to complement the coverage of cellular networks, an architecture consisting of dual, cellular and Wi-Fi equipped devices, simultaneously operating in cellular and ad-hoc mode, has been proposed in [1] to improve the downlink rates of cellular clients. The model replaces direct cellular connections with freshly established paths of relayers whose cellular rates improve upon the rates of the cellular clients. Since wireless LANs offer high throughput (IEEE 802.11b offers up to 11Mbps), albeit in a range of less than 200m, using a web of multihop paths can considerably increase the throughput from the base station to the devices in its cell without requiring modifications in the infrastructure.

The advantage of using a dual interface is that a multihop path from a host to the base station can have a better downlink rate than the direct connection of the same host to the base station. Also, the presence of a permanent link offers possibilities for efficient routing that are not available to ad hoc networks. The cellular interface has a low capacity, but if used intelligently, it can reduce the complexity of routing.

A simple solution to the problem of multihop path discovery in a hybrid network is also described in [1]. An initiator discovers a path to the base station with a breadth-first search of the network. The disadvantage of flooding the network every time a path is needed is that the traffic generated by the routing protocol can cause severe congestion. Another disadvantage is that when multiple hosts try to find a path to the base station, hosts that have a good downlink rate will be congested as they will be on many paths. The routing protocols of [1] do not take into account the possibility that links have to be shared among multiple paths. Also, the family of 802.11 Wi-Fi standards offers multiple, non-overlapping channels, which the protocol of [1], like most routing protocols, does not take advantage of. We offer a simple heuristic for utilizing multiple channels and avoid interference.

To overcome these problems, we propose a routing algorithm that is based on the maintenance of a spanning tree of the network. There are two reasons for using a spanning tree. The first is that, as we will show, a maximum spanning tree provides the optimal routing for the next flow from a host to the base station that has to be scheduled. We do not maintain a maximum spanning tree, but our protocol will lazily converge to one. Experiments show that the achieved throughput is consistently over 80% of the optimal.

The second reason for structuring the routing information as a spanning tree is that we can maintain it by generating $\mathtt{O}(\log \mathtt{n})$ traffic for each routing request, where $\mathtt{n}$ is the number of nodes in the network, instead of $\mathtt{O}(\mathtt{n})$, when flooding is used. We achieve this with *topology trees* [2], an example of link-cut trees [3]. The entire protocol has two layers. The first operates on the spanning tree, by issuing queries and update requests. The second implements these operations on the topology tree. Section 3 describes how the two layers interface and the importance of each. Section 4 compares the performance of our protocol with UCAN and with the optimal achievable throughput. Section 5 places our contribution in the perspective of related work, and Section 6 presents our conclusions.

2 Network Model

We assume a wireless ad-hoc network of $\mathtt{n}$ hosts, all situated inside the coverage area of a single cellular base station. Each mobile host is equipped with a dual cellular and Wi-Fi network card. We assume that the cellular base station can support simultaneous transmissions to/from all the hosts in its coverage area. We model the existing network as a graph, where the mobile hosts and the base station are nodes and links denote wireless connectivity. Thus, there exists a link between the base station and each host. We consider only undirected links, since this is also an assumption of the underlying wireless MAC protocol. Each link $\mathtt{e}$ has a constant weight $\mathtt{w}(\mathtt{e})$ equal to its data capacity, thus ignoring the time variation of the channel strength due to multipath fading.

3 Distributed Spanning Tree (DST) Algorithm

3.1 Interference and Aggregate Throughput

Consider a hybrid network where no host needs a multi-hop path to the base station (see Figure 1(a)). All links are available to their full capacity and we assume that there is a method to estimate this capacity (for example, hosts periodically ping their neighborhood and measure the response time). If a host `A` needs the best available path to the base station(`BS`) and there is an optimal path discovery protocol, `A` can find this path and establish a flow to the base station. Let this path be `A`, `B`, `C`, `D` and the capacity of the links be 11Mbps for $(\mathtt{B},\mathtt{A})$, 5.5Mbps for $(\mathtt{C},\mathtt{B})$, 11Mbps for $(\mathtt{D},\mathtt{C})$ and 2.1Mbps for $(\mathtt{BS},\mathtt{D})$. As mentioned in Section 2), the transmission between BS and `D` does not interfere with the ad-hoc transmissions, hence, the aggregate throughput of the path is the minimum between the capacity of $(\mathtt{BS},\mathtt{D})$ and the aggregate throughput of the ad-hoc path between `D` and `A`. Moreover, since each host is equipped with a single transceiver (see Section 2), the transmissions on $(\mathtt{B},\mathtt{A})$ and $(\mathtt{C},\mathtt{B})$ and also the transmissions on $(\mathtt{C},\mathtt{B})$ and $(\mathtt{D},\mathtt{C})$ cannot proceed simultaneously. Note also that the transmission between `D` and `C` interferes at `C` with the transmission between `B` and `A`. This is called *self-interference*. Thus, the aggregate throughput of the ad-hoc path between `D` and `A` is only a fraction (one third) of the capacity of the bottleneck link, $(\mathtt{C},\mathtt{B})$.

In our example, the capacity of the path is 1.8Mbps. This means that `A` can receive data on this path at a rate decided by the capacity of the $(\mathtt{C},\mathtt{B})$ link (see Figure 1(b)). This leaves links $(\mathtt{BS},\mathtt{D})$, $(\mathtt{D},\mathtt{C})$ and $(\mathtt{B},\mathtt{A})$ with a residual capacity. We take a conservative approach and block any transmissions on these links for the duration of the flow introduced by `A`. In addition, transmissions on links of hosts adjacent to the flow path also interfere with the flow. For example, a transmission on link $(\mathtt{G},\mathtt{F})$ interferes at `F` with the transmission on link $(\mathtt{B},\mathtt{A})$ and a transmission on $(\mathtt{F},\mathtt{E})$ interferes at `B` with a transmission on $(\mathtt{C},\mathtt{B})$. We conservatively model the interference introduced on links of hosts adjacent to the flow path by blocking transmissions on them for the duration of the flow. As a result, after each time a flow is added or removed, we can deduce the state of the residual network from the physical state of the links and the sequence of flow additions and deletions.

3.2 Multiple Channels

Our approach of modeling the residual capacity of links due to the addition of flows and the interference they introduce is conservative and may reduce the network throughput. To overcome this problem, we take advantage of the multi-channel capabilities of the 802.11b and 802.11a wireless standards. That is, interfering transmissions can be scheduled to occur simultaneously as long as they use non-overlapping frequencies. While 802.11b offers 12 transmission frequencies, only 3 can be chosen without overlappings. However, 802.11a provides 12 non-overlapping channels.

Choosing the transmission channel for a new flow can be done by a simple traversal of the path. For each traversed link, reserve the first locally available channel. Then, contact all the hosts whose potential transmissions interfere with the link (hosts adjacent to the link's endpoints) and reserve the chosen channel. If a link of an interfering host is left without any available channels, its residual capacity becomes 0. Figure 1(c) shows an example of this approach,

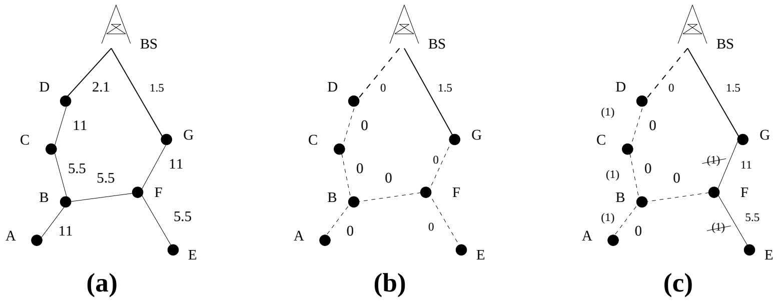


Figure 1. (a) Example of hybrid network, where labels on the right-hand side of links represent link residual capacities. (b) Residual network of (a) after A adds a flow on links $(\mathtt{BS},\mathtt{D})$, $(\mathtt{D},\mathtt{C})$, $(\mathtt{C},\mathtt{B})$ and $(\mathtt{B},\mathtt{A})$. Due to interference, not only links adjacent to this path but also links of hosts adjacent to this path are blocked. (c) Same scenario as in (b), only using the multi-channel capability of Wi-Fi standards. Labels on the left-hand side of links represent channel assignments. Links of hosts adjacent to the flow path, i.e. $(\mathtt{G},\mathtt{F})$ retain their capacity, but cannot use the channel chosen by A's flow due to interference.

where the ad-hoc links supporting the newly added flow of A reserve channel 1 and subsequently, channel 1 becomes unavailable for transmissions on links interfering with the flow. However, the available capacities of the interfering links are left unaltered.

3.3 Spanning Trees

A routing protocol that maintains the optimal path for each host in the residual network can discover multihop paths by only keeping a parent pointer for each host. In our example, the parent of A is B, the parent of B is C, the parent of C is D and the parent of D is the base station. After A adds a flow, the parent information might have to change to reflect the decrease in the capacity of the path links. At all times, the routing information constitutes a spanning tree rooted at the base station.

Using the cycle property [4] of maximum spanning trees, it can be proved that the maximum[1] spanning tree of a residual network provides the optimal routing information. The path from each host to the root in the spanning tree has the maximum minimum link possible. Since the aggregate throughput of a path is a fraction of the capacity of the bottleneck link of the path, this is guaranteed to maximize the capacity of the entire path.

Given a maximum spanning tree, a host can schedule the next flow by sending a forward request to its parent, which in turn will forward the request to its parent, until the base station is reached. The problem is after adding (or deleting) a flow, the entire network may need to be contacted to derive the new maximum spanning tree, which is asymptotically not better than flooding each time a flow needs to be scheduled.

To solve the scalability issues, each time there is a change in the network, we can lazily converge to the maximum spanning tree, instead of trying to keep up with the changes. Our distributed spanning tree (DST) protocol does not change the routing information to correspond to the maximum spanning tree each time a flow is added or removed. However, each time a host requests a path, its parent pointer is set to the neighbor that has the optimal path to the base station, according to the existing information. The changes are confined in the neighborhood of a host and while queries about the state of the paths of the neighbors have to contact hosts outside the neighborhood, they can be completed much faster, as we will show. We note that if the networks becomes static, the routing information will eventually converge to the optimal, even with this localized updating policy.

The advantage of this approach is that efficiently maintaining a spanning tree is possible even for large networks, as it requires only $\mathtt{O}(\log \mathtt{n})$ time and messages for each operation, the complexity of maintaining a dynamic tree. In the worst case, it can be arbitrarily far from the optimal, but our experiments indicate that, on average, the throughput achieved is not far from the optimal. For large and active networks, where nodes request flows frequently and links are close to capacity, DST performs extremely well.

3.4 Maintaining a Dynamic Tree

As we have mentioned, the entire routing protocol can be split into two layers communicating through a well-defined interface. The top layer is responsible for the maintenance of the distributed spanning tree. This layer issues a string of operations on the spanning tree. The following lists the operations:

---

[1] We talk about maximum spanning trees because we want to maximize the capacity of a path. If we define a cost metric, we should instead talk about minimum spanning trees.

- $\mathtt{Link}(\mathtt{v}, \mathtt{u}, \mathtt{w})$ Merge the tree rooted at node `v` with the tree of node `u` by making `u` the parent of `v`. The weight of the new link is `w`.
- $\mathtt{Cut}(\mathtt{v})$ Split a tree into two by removing the link of node `v` to its parent.
- $\mathtt{Mincost}(\mathtt{v})$ Return the minimum weight cost edge on the path from node `v` to the root of the tree it belongs to.
- $\mathtt{Update}(\mathtt{v}, \mathtt{w})$ Add `w` to all edges on the path from `v` to the root of its tree.

The second layer is responsible for efficiently completing these requests. We have chosen to implement this layer with a link-cut tree. Link-cut trees are structures that can complete the above described set of operations in $\mathtt{O}(\log \mathtt{n})$ time, where `n` is the number of hosts. This scalability property is important as it translates in $\mathtt{O}(\log \mathtt{n})$ messages when implemented distributively. Furthermore, when a new host enters the network and has to query its neighbors on the capacity of their paths, the parallel time complexity is $\mathtt{O}(\log \mathtt{n})$. For implementation purposes we have chosen to use topology trees for this layer. A topology tree is a relatively simple link-cut tree and it has a natural distributed implementation. We discuss topology trees and how the interface to this layer is implemented in more detail in Section 3.6. We note that in principle any dynamic tree can be used for this layer, as long as it does not modify the structure of the spanning tree. The root of the tree is fixed to be the base station and the links are oriented towards it. Balancing operations may have to change the root and the orientation of the links and cannot be used.

3.5 Refresh Rate

In dynamic networks, routes become stale quickly. A parent pointer indicating the best available path should be reevaluated at constant intervals to adapt to topology changes. The exact refresh rate depends on how dynamic the hybrid network is and how much traffic per operation we want to allow. There are two possible strategies on reassessing the parent pointer. The first is more aggressive, but generates more traffic. A node can cut its parent every `k` seconds and probe all its neighbors. The traffic generated is $\mathtt{O}(\mathtt{d} \log \mathtt{n})$, where `d` is the number of neighbors, but the parent pointer is as close to the optimal as possible with the available information. The second, less expensive, strategy is to query the parent every `k` seconds and cut it only if the rate falls below a threshold. We have used in our experiments the first strategy.

To keep the number of messages per time unit at a scalable level, we have to modulate `k` with the size of the network. As the network becomes more dense, the refresh rate should drop. If $\mathtt{k} = \Theta(\mathtt{d})$, the traffic generated every k seconds is $\mathtt{O}(\mathtt{k} \log \mathtt{n})$. The exact constants depend on the specifics of the network, but the conclusion is that for dense networks, refreshing should be done more sparingly. This appears to be contrary to the scalability of DST, as the spanning tree maintained should be farther from the optimal. However, dense hybrid networks are more robust, and even if a link disappears, there is a high probability that an alternate equivalent link will be present. A parent pointer can be used with relative confidence for the short time until the next reassessment. In Section 4, we use an alternate approach, scaling `k` with $\log \mathtt{n}$, which generates $\mathtt{O}(\mathtt{d})$ messages per time unit for each host. Even at this rate, the scalability of the network is not affected, as probing one's neighbors with a heartbeat broadcast, an operation performed by almost all wireless interfaces, generates `d` replies. Experiments confirm that dropping the refresh rate according to $\log \mathtt{n}$ does not affect the performance of DST.

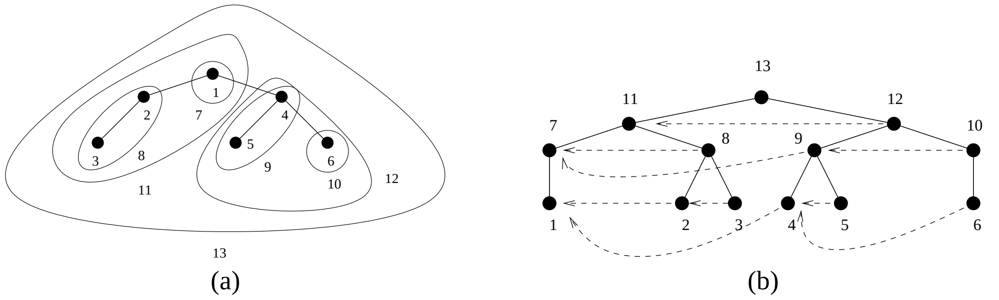
3.6 Topology Trees

The characteristic of topology and, in general, link-cut trees is that they can maintain dynamic trees of n nodes with $\mathtt{O}(\log \mathtt{n})$ operations per tree update. Also, operations that require an aggregate of all the nodes on a path from the root to a leaf, like `Mincost` or `Update`, can be completed in $\mathtt{O}(\log \mathtt{n})$ time. In fact, all operations of the interface to the second layer have $\mathtt{O}(\log \mathtt{n})$ time and message complexity.

The reason for choosing topology trees is that they are conceptually simpler than the more common splay trees. They are also naturally distributable structures and asymptotically they are optimal. Experimental results from [2] indicate that although they are less efficient than splay trees, the difference does not offset their implementation advantages. We will give a brief overview of topology trees in this section. For a detailed presentation and an example of topology trees supporting a complicated minimum spanning tree algorithm see [5].

Topology trees are derived from a restricted partition of a tree. For an example of a restricted partition see Figure 2(a). To avoid confusion, we will refer to nodes of the topology tree as clusters. The leaves of the topology tree are clusters of single spanning tree nodes. A cluster of a higher level is made up of one or two clusters of a lower level. The rules by which clusters are paired are described in [2]. The intuition is that for every pair of clusters that combines for a cluster of a higher level, another cluster is made up of a single lower level cluster to act as a buffer when there is an update in the structure of the topology tree. These buffers are clusters that have two children. In Figure 2(a), cluster 7 consists of only cluster 1 for this reason. It can be shown that the height of a topology tree is $\mathtt{O}(\log \mathtt{n})$ [2].

The resulting topology tree is shown in Figure 2(b). The



(a) (b)

Figure 2. (a) Example of a restricted partition and (b) the resulting topology tree.

solid edges indicate the relationship between clusters of consecutive levels. The dashed arrows represent the structure of the tree formed by clusters of the same level. Each such tree is called the induced tree of the specific level. The lowest level induced tree is the actual spanning tree. Observe that a cluster with two induced children will have only one child in the topology tree.

Besides the adjacency information, each cluster stores three more fields, `Δcost`, `nodemin` and `minvert`. If we want to calculate the weight of an edge from node `v` to its parent, we need to traverse the topology tree from the leaf cluster corresponding to `v` up to the root of the topology tree and sum the `Δcost` fields of the accessed clusters. The `nodemin` and `minvert` fields of a cluster `c` store information about the minimum cost edge in the spanning subtree induced by the leaves of the subtree rooted at `c`. The rules by which these fields are calculated and how the adjacency information should change after an update of the spanning tree can be found in [2] and in more detail in [5]. We note that each of the $\mathtt{O}(\log \mathtt{n})$ steps is a local operation that only needs information from the parent and sibling cluster to complete. This is important not only because it leads to $\mathtt{O}(\log \mathtt{n})$ total time for each of the operations of the topology tree, but also because it facilitates the distributed implementation of topology trees.

4 Simulation Results

In this section we present an experimental analysis of the throughput performance of DST with regard to the optimum throughput achievable when a centralized knowledge Bellman-Ford algorithm is executed. We model the ad-hoc network using the unit disk graph model, using the Agere Short Antenna PC Card Extended specification. We use the ARF [6] mechanism to establish the transmission rate of the communication channel between two mobile hosts. We use only the top two transmission rates, of 11Mbps for distances under 160m and of 5.5Mbps for distances under 270m. The hosts are initially deployed randomly in a square of area $2830 \times 2830\mathtt{m}^2$ and we use the random waypoint model [7] to simulate their movements. That is, each host chooses a destination point inside the deployment square, and moves toward it with a speed chosen randomly between 1 and the maximum speed. After reaching the destination, the host chooses a new destination and a new random speed. We have overestimated the effects of ad-hoc link interference, by blocking any transmission involving hosts adjacent to the link.

We use the UCAN [1] approach to model the dependency between the cellular link rates of hosts and their distance from the cellular base station. The base station is positioned at the center of the $2830 \times 2830\mathtt{m}^2$ deployment square and its cellular transmission range is 1920m. According to this model, each host inside the square is covered by the cellular transmission range of the base station.

We model the optimal throughput to be the one achieved by running the Bellman-Ford algorithm. Instead of computing the shortest path, we compute the maximum throughput path between the base station and all the mobile hosts it covers. In the case of multiple concurrent flows the optimal for `n` flows is computed by running Bellman-Ford on the residual network obtained after removing the bandwidth consumed and the interference introduced by the first $\mathtt{n} - 1$ flows.

We perform each experiment by choosing 5 different initial network configurations. For each such configuration the experiment is run for 100 seconds. Thus, each point on the plots is an average over 500 measurements.

In addition to the performance of DST when a single transmission channel is used, we also experiment with multi-channel transmissions. First, we use the 3 non-overlapping channels of 802.11b. Secondly, we switch to the 802.11a specification, providing 12 non-overlapping channels and transmission rates of up to 54Mbps. However, for our simulations we use only the 54Mbps and 48Mbps links.

In the first experiment we randomly deploy 300 hosts that continuously move with a maximum speed of 9m/s. We increase the number of simultaneously supported flows

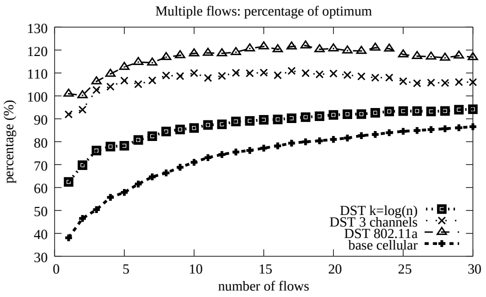


Figure 3. The throughput of DST and the basic cellular rate as percentage of the throughput achieved by Bellman-Ford when the number of flows grows from 1 to 30 for a network of 300 hosts.

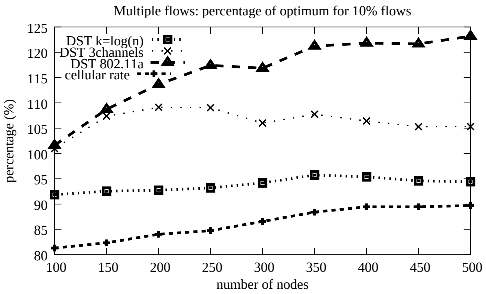


Figure 4. The throughput of DST and the basic cellular rate as percentage of the throughput achieved by Bellman-Ford for networks of 50 to 500 hosts, when 10% of the hosts concurrently hold a flow.

from 1 to 30. Figure 3 shows the performance of DST relative to the optimal total throughput, achieved when all the client hosts run the distributed Bellman-Ford algorithm to find the best downlink path. The performance of DST increases to achieve more than 90% of Bellman-Ford. In addition, Figure 3 also shows the performance of DST when using the multi-channel capabilities of 802.11b and 802.11a, compared to the flat cellular rate of the client hosts. Using the non-overlapping channels of 802.11b brings an increase of around 10% over the optimum achievable in the case of a single channel, while 802.11a has a 20% increase. Note that even when 30 out of the 300 hosts concurrently support a flow, by using the 3 channels of 802.11b, DST achieves a per-flow increase of 200kbps over the basic cellular rate.

The second simulation experiments with increasing concentrations of mobile hosts and of concurrent flows. In the same square area of $2830 \times 2830\mathtt{m}^2$, we place between 100 and 500 hosts, while also increasing the number of hosts concurrently supporting flows to be 10% of the total number of hosts. Figure 4 shows that DST performs very close to the Bellman-Ford, always higher than 90%. Using the 3 channels of 802.11b brings a 10% increase over the single channel variant, whereas using 802.11a achieves a throughput increase of up to 25% over the optimum Bellman-Ford. While the basic cellular rate remains constant, as the network becomes congested, the throughput achieved by DST per flow gracefully decreases when using the single channel or the multi-channel capabilities of 802.11b. However, when using DST in conjunction with 802.11a, the throughput per-flow saturates at 1050Kbps. This is because the usage of multiple non-overlapping channels alleviates the effects of the congestion generated at the hosts situated in the vicinity of the base station, by allowing concurrent transmissions on their adjacent hosts. Using DST with the 3-channel variant of 802.11b, brings an increase of between 150 and 200kbps over the cellular throughput. When using DST in conjunction with 802.11a, the throughput increase is more substantial, between 200 and 300kbps.

5 Related Work

The most popular model of wireless networks in the literature is that of the ad hoc architecture [8], [9], [7]. The entirely distributed nature of ad hoc networks limits their scope, as maintaining a connected network over a large area is quite difficult. There have been efforts to integrate infrastructure-based network models with ad hoc components, but most of them assume single-interface devices. In [10], GSM terminals are used to relay information to other terminals to improve coverage. In Opportunity Driven Multiple Access [11], transmission power is conserved by relaying traffic from a CDMA host to the base station through multiple, short hops. In [12], some channels are reserved for forwarding when the fixed channels become congested. In [13], a generic wireless network is considered, where hosts contact a mobile base station for access outside their cell, using only one interface. In [14], a hybrid network using the IEEE 802.11 architecture with both DCF and PCF modes is examined, using only one wireless interface. In [15], multihop paths are used to decrease the number of base stations by increasing their coverage. The overall capacity increases only when two communicating hosts are in the same cell.

Although double-interface architectures are conceptually similar to their single-interface counterparts, they increase the overall capacity by using short-range, high-bandwidth, ephemeral channels to relay traffic and a long-range, low-bandwidth, permanent channel to complete operations like routing and data integrity confirmation or as a last resort in the absence of neighbors. The low-bandwidth channels are not necessarily cellular, but the already existing infrastructure make them attractive options. This architecture has been examined in [1]. In [16], traffic is diverted to neighboring cells to increase throughput. The use of dedicated, stationary relays increases the cost of their solution and limits its utility. A study of local area hybrid networks is in [17]. A comprehensive presentation of a rudimentary hybrid network can be found in [18].

The problem of maintaining dynamic spanning trees is well-studied. In the context of ad hoc networks we are interested in the complete dynamic model, where hosts can turn on and off arbitrarily, in addition to edge deletions and insertions. This is the most powerful model of dynamic networks. In [19], a fully dynamic minimum spanning tree is maintained in $\mathtt{O}(\mathtt{n}^{1/3}\log\mathtt{n})$ time per update. We note that topology trees can be used to maintain a minimum spanning tree in $\mathtt{O}(\sqrt{\mathtt{m}})$ time, where m is the number of edges. For planar graphs, maintaining a minimum spanning tree is more efficient and can be achieved with $\mathtt{O}(\log\mathtt{n})$ time per update [20].

6 Concluding Remarks

We have described and evaluated analytically and experimentally the DST protocol for discovering multihop paths in hybrid networks. The strength of DST is its scalability. In metropolitan areas, where a cell may need to serve hundreds of mobile hosts requesting Internet access, it is crucial that routing has a low time and message complexity and that its performance does not suffer as the network size increases. DST exhibits all these characteristics, by maintaining a spanning tree of the network that is close to optimal, but without the overhead of being exactly the optimal. By using topology trees to maintain the dynamic spanning tree, each operation can be completed in $\mathtt{O}(\log\mathtt{n})$ time, generating $\mathtt{O}(\log\mathtt{n})$ messages. The total throughput is constantly close or over 80% of the optimal routing for active networks. Solutions relying on flooding are unscalable regarding both complexity and performance, as our experiments indicate.

References

[1] Haiyun Luo, Ramachandran Ramjee, Prasun Sinha, Li (Erran) Li, and Songwu Lu. Ucan: a unified cellular and ad-hoc network architecture. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 353–367. ACM Press, 2003.

[2] G. N. Frederickson. A data structure for dynamically maintaining rooted trees. In *SODA*, 1993.

[3] D. D. Sleator and R. E. Tarjan. A data structure for dynamic trees. *J. Comput. Sci.*, 26:362–391, 1983.

[4] Robert Endre Tarjan. *Data structures and network algorithms*. Society for Industrial and Applied Mathematics, 1983.

[5] G. N. Frederickson. Ambivalent data structures for dynamic 2-edge-connectivity and k smallest spanning trees. *SIAM J. of Comp.*, 26(2):484–538, 1997.

[6] A. Kamerman and L. Monteban. Wavelan-ii: A high performance wireless lan for the unlicensed band. *Bell Labs Technical Journal*, 1997.

[7] D.B. Johnson and D.A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing, volume 353, pages 153-181. Kluwer Academic Publishers*, 1996.

[8] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the 4th annual international conference on Mobile computing and networking*, pages 85–97. ACM Press, 1998.

[9] C.E. Perkins and E.M. Royer. Ad-hoc on demand distance vector routing, 1999.

[10] G.N. Aggelou and R. Tafazolli. On the relaying capacity of next-generation gsm cellulat networks. February 2001.

[11] T. Rousse, I. Band, and S. McLaughlin. Capacity and power investigation of opportunity driven multiple access (odma) networks in tdd-cdma based systems. 2002.

[12] X. Wu, S.-H. Chan, and B. Mukherjee. Madf: A novel approach to add an ad-hoc overlay on a fixed cellular infrastructure. In *Proceedings of IEEE WCWN*, 2000.

[13] I. Akyildiz, W. Yen, and B.Yener. A new hierarchical routing protocol for dynamic multihop wireless networks. In *Proceedings of IEEE INFOCOM*, 1997.

[14] H.-Y. Hsieh and R. Sivakumar. On using the ad-hoc network model in wireless packet data networks. In *Proceedings of ACM MOBIHOC*, 2002.

[15] Y.-D. Lin and Y.-C. Hsu. Mulithop cellular: A new architecture for wireless communications. In *Proceedings of IEEE INFOCOM*, 2000.

[16] S. De, O. Tonguz, H. Wu, and C. Qiao. Integrated cellular and ad hoc relay (icar) systems: Pushing the performance limits of conventional wireless networks. In *Proceedings of HICSS-37*, 2002.

[17] S. Lee, S. Banerjee, and B. Bhattacharjee. The case for a multi-hop wireless local area network. In *Proceedings of IEEE INFOCOM*, 2004.

[18] M.J. Miller, W.D. List, and N.H. Vaidya. A hybrid network implementation to extend infrastructure reach. Technical report, University of Illinois at Urbana Champaign, January 2003.

[19] Monica Henzinger and Valerie King. Maintaining minimum spanning trees in dynamic graphs. In *ICALP*, 1997.

[20] D. Eppstein, G. F. Italiano, R. Tamassia, R. E. Tarjan, J. Westbrook, and M. Yung. Maintenance of a minimum spanning forest in a dynamic planar graph. In *SODA*, 1990.