A New Paradigm for Load Balancing in Wireless Mesh Networks

Abstract: *Obtaining maximum throughput across a network or a mesh through optimal load balancing is known to be an NP-hard problem. Designing efficient load balancing algorithms for networks in the wireless domain becomes an especially challenging task due to the limited bandwidth available. In this paper we present heuristic algorithms for load balancing and maximum throughput scheduling in Wireless Mesh Networks with stationary nodes. The goals are to (a) improve the network throughput through admissibly optimal distribution of the network traffic across the wireless links, (b) ensure that the scheme is secure, and (c) ensure fairness to all nodes in the network for bandwidth allocation. The main consideration is the routing of non-local traffic between the nodes and the destination via multiple Internet gateways. Our schemes split an individual node's traffic to the Internet across multiple gateways that are accessible from it. Simulation results show that this approach results in marked increase in average network throughput in moderate to heavy traffic scenarios. We also prove that in our algorithm it is very difficult for an adversary to block a fraction of a node's available paths, making it extremely hard to compromise all traffic from a node. Simulation results also show that our scheme is admissibly fair in bandwidth allocation even to nodes with longest paths to the gateway nodes.*

Key Words: Fairness, Load Balancing, Resiliency, Traffic Splitting, Throughput, Wireless Mesh Networks

1. Introduction

Optimal load balancing across a mesh or a network is a known hard problem. [1] Describes load balancing as an optimization problem. [2], [3], [4], [5], prove the NP-completeness of various load-balancing problems and provide approximation algorithms. Efficient load balancing in wireless networks becomes an even more challenging problem due to the limitations on available bandwidth and unreliability of wireless links.

In this paper we consider load balancing in wireless mesh networks with stationary nodes. These include Wireless Mesh [6], [7] and Community networks [8], such as the Self-Organizing Neighborhood Wireless Mesh Networks [9]. We provide heuristic algorithms for load balancing in wireless mesh networks with the following goals: (a) maximize network throughput through admissibly optimal distribution of the network traffic across the wireless links, (b) ensure the scheme is secure, and (c) ensure fairness to all nodes in the network for bandwidth allocation. Below we first provide motivation for singling out these networks over other conventional wireless networks in our study for designing load balancing heuristic algorithms.

1.1. Motivation

The stationary nature of the nodes in wireless mesh networks warrants the design of robust and efficient traffic management protocols for them. Existing single path and multipath traffic protocols for wireless networks are designed for single source-destination pairs (see Sec. 2). Mesh and community network nodes, by contrast, can have accessibility to multiple Internet gateways connecting them to the Internet backbone and each node can individually select the best gateway for its non-local traffic (traffic to the Internet). Since the bulk of node traffic in such networks would be non-local, this will lead to performance and fairness issues for the network. Load balancing in wireless mesh networks is an interesting and unique problem different from conventional wireless networks due to several reasons:

- **Nodes are stationary**. Existing wireless ad-hoc network protocols are designed with node mobility considerations. Sensor network protocols have power and computation constraints. Thus, neither ad-hoc, nor sensor network protocols would be suitable for mesh networks. Better network performance is in order through protocols designed specifically for mesh networks with stationary wireless nodes.
- **Multiple gateways are available**. Load balancing can be improved by splitting network traffic across the accessible gateways and reassembling this traffic on the Distribution System (wired backbone network). This may be a better approach incase of some wireless links getting loaded to capacity and others being underutilized. This is akin to IP [10] routing for the wired networks where different packets between a source and destination may be routed along different paths.

Clearly, above conditions are different from ad-hoc and sensor networks and require designing a traffic distribution protocol specifically for this scenario.

1.2. Summary of Contributions

This paper proposes a new traffic distribution and load balancing protocol for stationary mesh networks. The focus is on traffic between the nodes and the backbone Internet. We show that optimal load balancing and maximum throughput scheduling for this scenario is NP complete through a simple reduction of the known NP complete Knapsack [11] problem. We present a heuristic algorithm for load balancing which aims at maximizing network throughput through efficient wireless-links utilization. The algorithm splits a node's non-local traffic across the multiple gateways connected to it. The rationale is that under heavy traffic load, if each node is able to send part of its traffic on its best available route, link utilization will be uniform throughout the network and average network performance will improve. We demonstrate the efficiency of our algorithm through simulations, evaluating it against other popular approaches.

In addition, single path node failures can be better dealt with and network robustness and resiliency against attacks can be enhanced through our scheme. We demonstrate that the problem of an adversary compromising a subset of paths from a node (by compromising some intermediate nodes on those paths, with each node having an associated cost to compromise it), such that the cost is less than some value K, is NP-complete. Finally, through simulations, we demonstrate that our scheme is admissibly fair in bandwidth allocation even to nodes with longest paths to the gateway nodes. Thus, our scheme achieves three goals (a) efficient load balancing, (b) security, and (c) fairness. To the best of our knowledge, this is the first scheme to consider load-balancing by simultaneously splitting a node's traffic to all available destinations gateways.

1.3. Paper Organization

This paper is organized as follows. Section 2 describes related work and gives an overview of our model. Section 3 presents proof of NP completeness of maximum throughput scheduling for mesh networks and details our heuristic load balancing algorithms. Section 4 talks about the security of our scheme. Section 5 demonstrates the fairness of our scheme and presents applicability analysis. Simulation results are presented in Section 6. Finally, in Section 7 we conclude the paper with a discussion of its limitations and proposed future work.

2. Related Work and Model Overview

2.1. Related Work

Conventional multi-hop wireless network traffic protocols are either single path [12], [13] or multipath [14]. In [15] the authors present a multipath load-balancing protocol for mobile ad-hoc networks with directional antennas for *maximally zone disjoint* routes. Ganjali et al. [16] compare load balancing in ad-hoc networks for single path and multipath routing. Multipath protocols maintain multiple paths, but use only one path at a given time. Only [17] researches simultaneous activation of multipaths, but in their study a packet randomly chooses one of several available paths. In addition, they consider multiple paths between single source-destination pairs. Our protocol has a well defined (non-random) algorithm for splitting a node's traffic and we consider simultaneous invocation of multiple paths between a node and multiple Internet gateways.

Mobile Mesh protocol [18], [19] describes schemes for link discovery, routing and border discovery in wireless mesh networks, but does not consider load balancing. In [20] authors describe choosing a high throughput path between a source and a destination for community wireless networks. Raniwala et al. [21] discuss load-balancing in wireless mesh networks with nodes having multi-channel radios, but these radios would require multiple cards and antennas for each node and would be expensive to deploy. Hespanha et al. [22] formulate secure load balanced routing in networks as a zero-sum game between the designer of the routing algorithm and an adversary that attempts to intersect packets. They show that for some versions of the game, the optimal routing policies also maximize the throughput between the source and the destination node.

There is extensive literature on optimization problems on dynamic and static load balancing across meshes [23]. Optimal load balancing across meshes is known to be a hard problem. Akyildiz et al. [6] exhaustively survey the research issues associated with wireless mesh networks and discuss the requirement to explore multipath routing for load balancing in these networks. However, maximum throughput scheduling and load balancing in wireless mesh networks is an unexplored problem.

In this paper we present for the first time, a load balancing scheme in wireless mesh networks by systematically splitting each node's non-local traffic to available destination gateways and evaluate its performance.

2.2. Assumptions and Model Overview

We consider a wireless mesh network model with stationary nodes. An example is a community network where buildings with wireless antennas mounted on them are the network nodes (Fig. 1) [9]. Nodes connect to the backbone Internet via gateways located in the community. Traffic is forwarded to the gateways by the nodes in a hop-by-hop fashion. Thus, each node acts as both transmitter and router. Traffic across various gateways in a region can be effectively reassembled at the Distribution System. Since the network is stationary, route changes are infrequent, and occur only incase of node failures or faults. As such, the network controller or a similar entity on the Distribution System maintains complete network information including the network topology, and can perform static route computation from any node to the gateways. Since this is an administered network, the network controller has an estimate of the upper limit on bandwidth required by each node for its self-traffic. Nodes are assumed to be non-malicious and cooperate in routing others traffic.

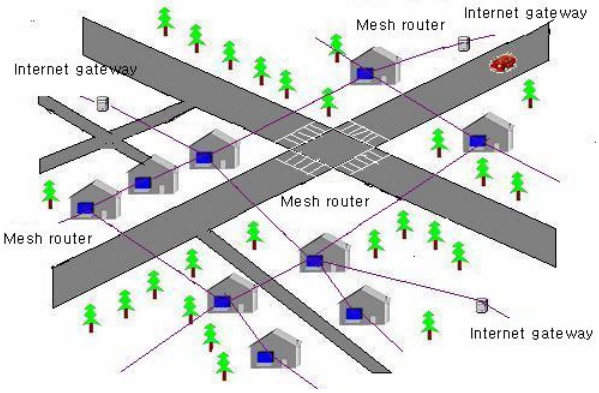

Fig. 1. Community Network

3. Maximum Throughput Scheduling

The primary objective of a Load Balancing scheme is to achieve maximal network throughput through uniform link utilization. The other goals include fairness to the nodes and robustness against attacks and node failures. Thus maximal throughput scheduling forms a subset of the larger Load Balancing problem. In this section we first show that the decision version of the maximum throughput scheduling optimization problem is NP-complete. We then present two heuristic algorithms aiming at maximum network throughput scheduling for load balancing. We call them Scheduling Schemes 1 and 2.

3.1. The Maximum Throughput Scheduling Problem

The maximum throughput scheduling optimization problem can be defined as follows. Suppose in a graph every node has some bit rate traffic to send with some specified paths (maybe multiple paths to the same destination), and every link has an upper bound on capacity, what kind scheduling (i.e., the order to reserve bandwidth for the nodes) can achieve maximum throughput? The corresponding decision problem is: is there a schedule such that the overall throughput of the network is greater than K?

Theorem 1 The decision version of the maximum throughput scheduling problem is NP-complete.

Proof: As a special case, consider a star network with center C. All nodes except for one destination node need to send traffic to the destination node via the center C. Now C needs to schedule all its received traffic to forward to the destination. This is a case of knapsack problem where one can consider the capacity of the link between C and destination as the capacity of the knapsack. Knapsack problem is NP-complete [11], so the maximum throughput scheduling problem is NP-complete.

3.2. Scheduling Scheme 1

This scheduling scheme computes the volume of traffic each node can send along its routes to connected gateways. The scheme requires the invocation of the Traffic Distribution Algorithm (TDA) shown in Fig. 2 (explained in Sec. 3.2.2). This algorithm computes traffic distribution for each node ahead of the actual routing based on link weights and node priorities. Any changes in the network topology (due to node failures or new nodes being added) would require a rerun of the algorithm.

Each round of TDA requires partial re-computation of the *current* shortest path from each node to its connected gateways. Instead of computing *the* shortest path from each node to all of its connected gateways, a minimum spanning tree is constructed rooted at each gateway. This gives the average shortest path from each gateway to the accessible nodes. This is an admissible approximation which simplifies the

computation and improves the running time. Below we first present the notations used in TDA and then describe the algorithm in detail.

3.2.1 Notations

- N : Set of network nodes
- t_{ix} : The traffic sent on link i by node x.
- T_x : Total traffic to be sent by node x in kbps.
- r_x : Number of routes from node x to its connected gateways, i.e., the number of gateways node x is connected to.
- P_n : Priority of node n. Equivalent to traffic routed by n for other nodes in kbps. For example, if node n routes 30 kbps traffic for another node, its priority becomes 30.
- RT_n : Remaining traffic to be sent by node n.
- C_j : Cost of link j. Equivalent to traffic routed on link j in kbps. For example, if link j is reserved for routing 30 kbps traffic, its cost becomes 30.
- D : $Max(r_x)$ for x = 1 to $|N|$. This is the maximum number of gateways that any node in the network has paths to, from the set of all nodes. This fixes the number of iterations of the algorithm.
- Z : the array containing $RT_n + P_n$ values for all nodes n belonging to N.
- S_x : The current shortest path to a gateway node from node x.

3.2.2. TDA Description

The TDA works by assigning costs to links and priorities to nodes. The ratio of the total traffic a node has to send, to the number of gateway nodes it is connected to (T_i / r_i), is a metric for distributing the node's traffic to its connected gateways. We assume Constant Bit Rate (CBR) traffic. The algorithm maintains k-shortest paths [24] from each node to gateway nodes. It runs iteratively to ensure fairness to all the nodes. In each iteration, the node n with the next highest $RT_n + P_n$ value has its shortest path S_n assigned to it successively. If any link in this present shortest path S_n exceeds the link capacity c_i, that shortest path is not used and the next-shortest path is computed. Node n then routes (T_n / r_n) of its traffic along path S_n. After n has been assigned this route, the cost of all the links along this route is incremented by the amount of traffic they will be routing for n i.e., $|t_{in}|$. All the nodes which have not yet been assigned their shortest paths in this iteration, have to re-compute their shortest paths for this iteration, as the cost of links along their original shortest paths may have now increased. The cost of a path is the sum of the cost of each link along that path. In order to be fair and to reward an intermediate hop node (i.e., a node that is neither the source nor destination in this session) for routing traffic of other nodes, we introduce a priority metric P.

```
N← {all nodes}
L← {links}
S← {shortest paths}

procedure TrafficDsitribute (N, L, S) {
 t_ix = [ T_x / r_x ] ;
 for each n ε N {
   P_n = 0;   RT_n = T_n ;
 }
 for each l ε L {
   C_l = 1;
 }
 done_count = 0;
 for iterations = 1 to D {
   Sort ( Z );    //sort array Z in decreasing order
    new_count = |N| − done_count
    for x = 0 to new_count {

   Choose node n_x corresponding to Z [x];
    check =0;
    while (check ==0){
      Compute S_x // current shortest path for n_x
      for each i, s.t.  l_i ε { S_x }
      if( (c_i+ t_ix) ≥ c_i {
            check = 0;
            Remove present path from list of k-
shortest paths for node n_x .
       }
      else {
            check = 1;
        }
      }  // end of (while check==0) loop
     S_x → n_x ;    //Assign  n_x  its shortest path
     for each i , s.t. l_i ε { S_x } {
         c_i = c_i + t_ix ;
         RT_x = T_x − t_ix ;
          if ( RT_x = 0 )
            done_count = done_count + 1;
          for each k, s.t. n_k  ε { S_x } //nodes lying on
the path S_x
            P_k =  P_k +  t_ix ;
          for each n_q , s.t q is from x to ( |N|  −
done_count )
            Recompute S_x ;
     }
    }
```

Fig. 2. Traffic Distribution Algorithm

The priority of all intermediate hop nodes on a path S_n is incremented *for the next iteration*, by the amount of traffic they will be routing for source node *n*. At the beginning of each iteration, array *Z* is sorted to determine the ordering of nodes according to their current $RT_i + P_i$ values. The number of iterations is determined by the value *D*. The algorithm splits each node's traffic according to the number of gateway nodes it is connected to and favors the nodes with the highest $RT_i + P_i$ values in assigning the shortest paths in each iteration.

3.3. Scheduling Scheme 2

In this section we present an alternate simple Traffic Scheduling scheme involving less computation. This is a "greedy" scheduling scheme and would perform coarse grained scheduling and load balancing as compared to the TDA.

Consider a node that has paths to n gateway nodes. Let $h_1, h_2, h_3 \ldots, h_{i\ldots,} h_{n,}$ be the number of hops along routes to gateway nodes 1 to n. The traffic from a source node is distributed in inverse proportion to the number of hops along all its routes: Highest volume of traffic is sent along the route with fewest hops. Fraction of traffic sent along a route i (Ti) will be computed as:

$$Ti = \{ [(h_1h_2h_{3..} h_{i\ldots}h_n) / (h_2h_{3..} h_{i..} h_n + h_1h_{3..} h_{i\ldots}h_n + \ldots + h_1h_2h_{3..} h_{i\ldots}h_{n-1})] * (1/ h_i) \}$$

For example, consider a node which is connected to three gateway nodes and its distance in number of hops from these gateway nodes is 2, 3 and 4, respectively. So, it sends $12/26^{th}$ fraction of its traffic to the closest gateway node, $8/26^{th}$ fraction to the next gateway node and $6/26^{th}$ fraction to the farthest gateway node.

4. Security and Resiliency

In this section we show that our heuristic algorithms for maximum throughput scheduling achieve robustness and resiliency against attacks and node failures. The robustness and security of a traffic scheme can be measured by how effectively it can be compromised or attacked by an adversary. An adversary can disrupt functioning of the network by blocking traffic to the gateways via compromising en-route nodes or localized jamming. This is one of the chief reasons for maintaining disjoint or braided multipaths [25]. Intuitively, a scheme using multiple paths simultaneously should be more robust against such disruptions than multipath (maintaining multiple paths, using one path at any time) schemes, making it harder for an adversary to block all active paths from a node. This is proved below.

We consider a more stringent threat model which assumes that traffic from a node is compromised if a subset of the node's active paths is blocked or compromised. This is based on the principles of threshold cryptography [26], where a secret is compromised if more than some percentage of shares is compromised. In other words, a secret is correctly received by the recipient, only if more than a certain percentage of shares are accurately received.

The localized link jamming by an adversary scenario is similar to using link cuts [27] to attack Internet routing [28]. For path blocking through en-route node compromise we present an optimization problem called the Minimum Cost Blocking (MCB) Problem. The MCB problem pertains to a set of nodes in a network and a set of paths between the nodes, with an associated cost to compromise each node. It seeks to find the minimum cost for an adversary to compromise a subset of the nodes in a network such that a certain percentage of the network paths are blocked? Here, we present two instances of the MCB problem, a special case and the general case.

4.1. The MCB Problem: Special Case

This is a special case of the MCB problem. The optimization problem for the adversary can be defined as follows. Suppose in graph G (V, E), |V| = n and every node v_i in V has a cost c_i to be compromised. Suppose there are m paths $(P_1, P_2, \ldots\ldots\ldots, P_m)$ from some sources to some destinations. Some paths may have the same source and destination as other paths (i.e., multipaths exist in this source and destination pair). What is the minimum cost to compromise a subset of the nodes such that a certain percentage of the paths are compromised?

The corresponding decision problem is: Given graph G = (V, E) and every node v_i in V has a cost c_i to be compromised. There are m paths $(P_1, P_2, \ldots\ldots\ldots, P_m)$, and integers K and R. Is there a subset V' of V such that V' will block R out of the m paths and the total cost of nodes in V' is no greater than K?

Theorem 2 The special case of MCB problem is NP-complete.

Proof: If we consider every path $(P_1, P_2, \ldots\ldots\ldots, P_m)$ as an element, then every node in the graph G can be

considered to comprise of a subset of all paths. That is, all the paths on which the node is located, constitute that node's subset. For example a node v_i in V may have a subset $\{P_j, P_k, P_l\}$ if v_i lies on the paths (P_j, P_k, P_l). Now the problem to find the minimum weight subset of nodes that block a part of the paths is equivalent to weighted partial set cover problem, which is NP-complete [29]. So the special case of the MCB problem is NP-complete.

4.2. The MCB Problem: General Case

Here we present the general case of the MCB problem for the adversary: Suppose in graph G(V,E), |V| = k, and every node v_i in V has a cost c_i to be compromised. Suppose we have $m = \sum_{i=1}^{k} n_i$ paths:

$(P_{11}, P_{12}, \cdots, P_{1n_1}, P_{21}, P_{22}, \cdots, P_{2n_2}, \cdots, P_{k1}, P_{k2}, \cdots, P_{kn_k})$.

Here $(P_{i1}, P_{i2}, \cdots, P_{in_i})$ are paths originating from node i $(i = 1,2,...,k)$. What is the minimum cost to compromise a subset of the nodes such that a certain percentage of paths that originate from a node are compromised (if a path originates from a node, we say that the path belongs to that node)? That is, for every node i, $(i = 1,2,...,k)$, there is a number $R_i, 0 \le R_i \le n_i$, and at least R_i paths out of all paths belonging to this node (paths $P_{i1}, P_{i2}, \cdots, P_{in_i}$) are compromised? This is a typical optimization problem. The decision problem corresponding to it is:

Given graph G=(V,E) and cost of every node, and set of nodes in $m = \sum_{i=1}^{k} n_i$ paths, $P_{11}, P_{12}, \cdots, P_{1n_1}, P_{21}, P_{22}, \cdots, P_{2n_2}, \cdots, P_{k1}, P_{k2}, \cdots, P_{kn_k}$ and integers C and $R_i, 0 \le R_i \le n_i$, is there a subset of V' of V such that V' will block at least R_i out of the paths $P_{i1}, P_{i2}, \cdots, P_{in_i}$, for all $(i = 1,2,\cdots,k)$, and the total cost of nodes in V' is no greater than C?

Theorem 3 The general case of the MCB problem is NP-complete.

Proof: The problem is a general case of the partial set cover problem [29], which is NP complete. So the general case of MCB problem is NP-complete.

Since the special and general cases of MCB problem are NP-complete, it will not be easy for an adversary to block a certain percentage of a node's paths (or some percentage of any paths) in the network, making our scheme resilient against attacks and failures.

5. Node Fairness and Applicability

This section (a) discusses the fairness of Scheduling Scheme 1 for all nodes being able to reserve bandwidth for their self-traffic, (b) discusses fairness and performance issues of Scheduling Scheme 2, and (c) compares the running time and applicability of our schemes in qualitative terms. This section complements the Simulation results in Sec. 6 for a better understanding of the performance of our schemes.

5.1. Fairness to Nodes: Scheduling Scheme 1

In Scheduling Scheme 1, during each iteration of TDA, every node is allowed to reserve bandwidth for a fraction of its traffic. This has two outstanding consequences: (a) No node starves for bandwidth, and (b) This schedule doesn't result in bandwidth wastage for nodes with less self-traffic. Thus, Scheduling Scheme 1 results in maximal fairness to the nodes in terms of opportunity to reserve bandwidth.

Due to priority assignments for routing non-self traffic, nodes closer to the gateways (for example, nodes 1-hop away from a gateway: called G-1 nodes in the rest of this paper) may eventually end up getting higher priorities in the later rounds of TDA. This results in such nodes being able to reserve their shortest paths before other nodes in later rounds. Consequently, these nodes can achieve higher throughputs than say, a node that is situated farther away from all gateways. This is further demonstrated by simulation results in Sec. 6.

Nevertheless this difference in throughput is not an indication of lack of fairness. It is natural for nodes in different parts of the network to achieve non-uniform throughputs due to the difference in path-lengths. A node situated in the centre of a network and uniformly distant from all available gateways will have lower throughput than nodes closer to the gateways, regardless of the scheduling or route reservation schemes used. The fairness of Scheduling Scheme 1 is demonstrated by the simulation results in Sec. 6 where it outperforms other well known schemes in terms of throughput for such a node. The fairness is also demonstrated by the simulation graphs showing uniform link utilization in Sec. 6. Uniform throughput

for all networks nodes, if desired, is achievable by a slight modification to TDA: the G-1 nodes and other nodes close to the gateways need to be initialized to priority values lower than 0 as opposed to other nodes.

5.2. Fairness and Performance: Scheduling Scheme 2

Scheduling Scheme 2 is fair to nodes in terms of bandwidth scheduling in that it splits every node's traffic to available paths, not favoring any nodes over others for traffic scheduling. This may result in nodes having to compete for bandwidth, but the scheme does not provide any inherent advantage to some nodes over others.

Scheme 2 selectively loads shorter paths with more traffic, causing links along shorter paths being loaded to capacity, while other links in the network may be lightly loaded. It results in lower throughput for the network nodes when compared with Scheme 1, though Scheme 2 still outperforms other well known schemes (as shown by simulation results). Performance vs. simplicity tradeoff for Scheme 2 is discussed in Sec. 5.3.

The reduction in throughput for Scheme 2 is partly attributable to and more pronounced for G-1 nodes. For G-1 and other nodes in the vicinity of gateways, small proportion of their traffic may be routed on very long routes even though the nodes themselves are close to the gateways. Though a very small volume of traffic may be affected due to this factor, it helps in keeping the design of the scheme simple (see Sec 5.3). A variant of the scheme could be to not assign traffic to routes that are longer than some threshold.

5.3. Running Time and Applicability Analysis

Scheduling Scheme 2 has a much lower computation complexity and running time as compared to Scheme 1. TDA has a computation time of $O(N^3)$ where N is the number of nodes in the network. The algorithm has to perform $O(N^2)$ computations in each iteration and has to perform O(N) iterations in the worst case. Scheduling Scheme 2 has a worst case computation time of $O(N^2)$.

The higher computation time of Scheme 1 is inconsequential for scenarios like the community networks because all the computations are performed offline and beforehand by the network controller or a similar entity on the distribution system. There would be very little change in topology for such networks, thus the load balancing computations would be very infrequent. However, if the network is dynamic due to node mobility, node sleep-time, or frequent node failures, then Scheme 2 would be beneficial. Scheme 2 would be especially helpful in such scenarios where frequent route re-computations are required.

An inherent assumption for both the schemes is that the cost of re-assembling the nodes' traffic at the gateways (or the distribution system) is offset by the savings incurred in terms of network bandwidth utilization and network throughput increase. This is a valid assumption since the wireless media is a shared resource, whereas an arbitrarily powerful machine can be employed for reassembling node traffic on the distribution system.

6. Simulation Results

We conduct simulations to investigate the performance of our schemes for each of the following three objectives: (a) Maximum Throughput Scheduling, (b) Minimum Cost Blocking, and (c) Fairness to nodes. We do a self-contained evaluation of the algorithms, transparent of underlying network level issues other than the capacity of wireless links.

We develop two specific schemes called Single Shortest Paths (SSP) algorithm and Fixed Shortest Path (FSP) algorithm so that we can evaluate TDA and Scheduling Scheme 2 against some basic benchmarks. In the SSP algorithm, each node determines one shortest path to a gateway and sends all self traffic on this shortest path. In the FSP algorithm, each node splits traffic equally on all its paths to connected gateways. This is similar to Scheme 2, except that a node's traffic is equally split along its available routes. Further, there are two variants for Scheme 2 and the FSP algorithm: large load-node first (favoring nodes with high volume of self-traffic) and small load-node first (favoring nodes with less self traffic). In the large load first scheme, nodes with the largest loads (heaviest traffic) schedule traffic along all their paths before nodes with smaller loads. In small load first, the order of scheduling is reversed from nodes with least traffic to nodes with heaviest traffic. If any links along a node's path reach capacity, then no more traffic can be scheduled across those links, and the node has to use other paths.

Our simulation topology is 100 nodes evenly distributed over a rectangular area of sides 1000 meters

by 1000 meters. There are 4 gateways, one at each corner of the rectangle. There are 15 G-1 nodes. We used a pseudo-random function for placing the nodes in the rectangular area and generating the links between the nodes. Once generated, the topology was fixed. All data points are average of 100 runs. In the simulation graphs, we refer to Scheduling Scheme 2 as Algorithm 2 for space conservation.

6.1. Throughput Comparison

We first compare the throughput of TDA and Algorithm 2 with SSP and FSP. In the simulation graphs, X-axis represents the average of all nodes' traffic in kbps. Y-axis represents the percentage throughput (1 corresponds to 100% throughput).

Figures 3, 4, 5 present total network throughput for network link capacities of 50, 100 and 150 kbps respectively. It is evident that TDA outperforms all other schemes in terms of network throughput under conditions of heavy traffic (high load, low link capacity), moderate traffic, and low traffic (low load, high link capacity); establishing that it is far superior to other schemes for network throughput. This is because TDA dynamically adjusts shortest paths and node priorities.

No other algorithm clearly dominates others in all scenarios; however the performance of SSP algorithm is consistently poor at all times. Performance of Algorithm 2 with large load first is only marginally better as scheduling nodes with large loads first results in shortest paths getting loaded with a few nodes' traffic initially. Subsequently a large number of nodes are forced to route most of their traffic along generally longer routes due to a lot of links getting loaded to capacity early on.

However, Algorithm 2 with small load first is generally better than the FSP algorithm under constrained link-bandwidth conditions as seen in Fig. 3. With the small load first schedule, more nodes are able to schedule traffic on their paths of first-choice (links are loaded with less traffic per node initially) and shorter paths are able to carry traffic for higher number of nodes. Only a few nodes with heavy traffic may find links along their shortest paths loaded to capacity later in the schedule. Thus the number of nodes affected by link congestion will be less. With FSP, the nodes split their traffic equally, thus the advantage of small load-node first becomes less prominent.

Figures 6, 7 and 8 show the throughput of a G-1 node for link capacities of 50, 100 and 150 kbps respectively. As discussed in Sec. 5, the performance of TDA is exceptionally higher than other schemes because G-1 nodes attain high priorities for routing other nodes' traffic. The performance of Algorithm 2 (small load first) is also consistently better than BSP and SSP algorithms.

Figures 9, 10 and 11 present the throughput of a node located roughly in the center of the network for link bandwidths of 50, 100 and 150 kbps respectively. Nodes situated within a square of 300 meters by 300 meters in the center of the simulation topology were considered candidates. Again the performance of TDA is better than other schemes. Performance of FSP (smallest load first), Algorithm 2 (smallest load first) and SSP are comparable: the difference in path lengths between shorter and longer paths is less significant for a node almost uniformly distant from all gateways. This offsets the advantage that Algorithm 2 and FSP have over SSP.

6.2. Robustness

As discussed in Sec. 4, threshold cryptography schemes assume that a secret is correctly received if a certain percentage of the shares are delivered accurately. Here we test our schemes for two different cases: a secret is correctly received if (a) 2 out of 4 and (b) 3 out of 4 shares from a node are received at the gateways. Figures 12, 13 and 14 represent the percentage of nodes that successfully sent t out of n (here 2 out of 4 and 3 out 4) threshold traffic for link bandwidth upper bounds 50, 100, 150 kbps respectively. The X-axis in the plots represents the percentage of nodes and the Y-axis represents the average node load. Again it is evident from the figures that TDA is the more robust scheme in both the cases.

6.3. Fairness

Figures 15, 16 and 17 show average link utilization for link capacities 50, 100 and 150 kbps respectively. The average link utilization is substantially higher for TDA in all three scenarios. This indicates that the TDA is able to load the links more efficiently and uniformly than the other schemes, representative of homogeneous traffic distribution across the network and enhanced node fairness. Since all the paths are dynamically chosen, any unused link bandwidth may be utilized after other links get congested.

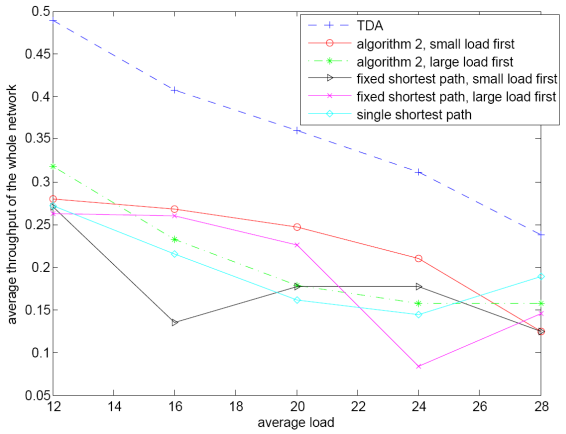

Fig. 3. Network Throughput (50 kbps)

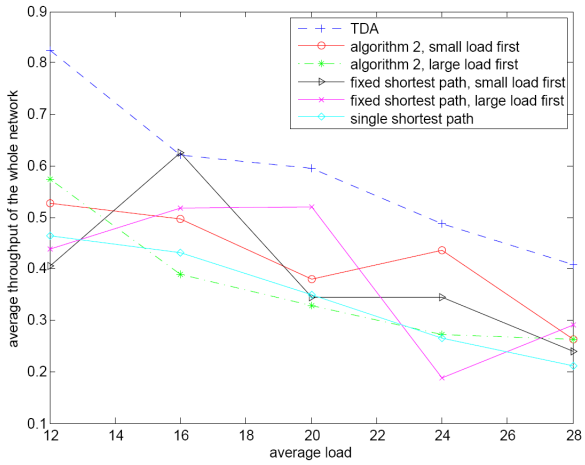

Fig. 4. Network Throughput (100 kbps)

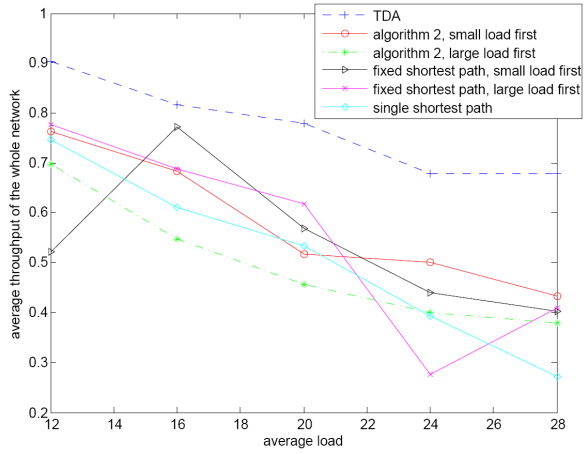

Fig. 5. Network Throughput (150 kbps)

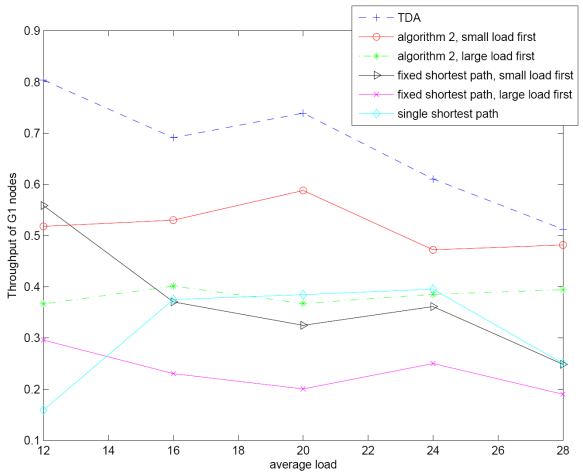

Fig. 6. G-1 Node Throughput (50 kbps)

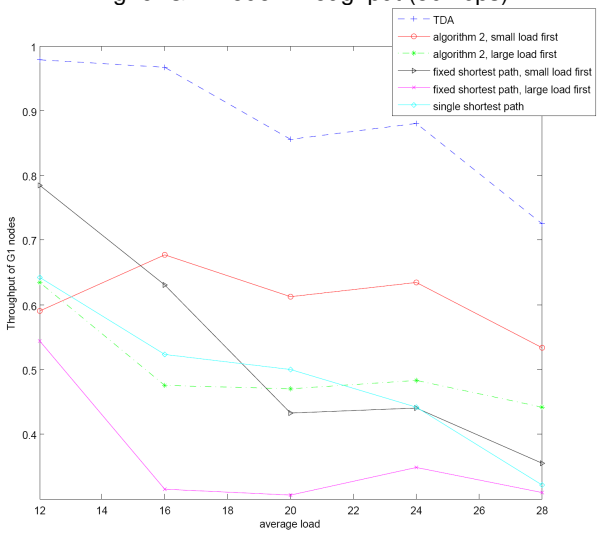

Fig. 7. G-1 Node Throughput (100 kbps)

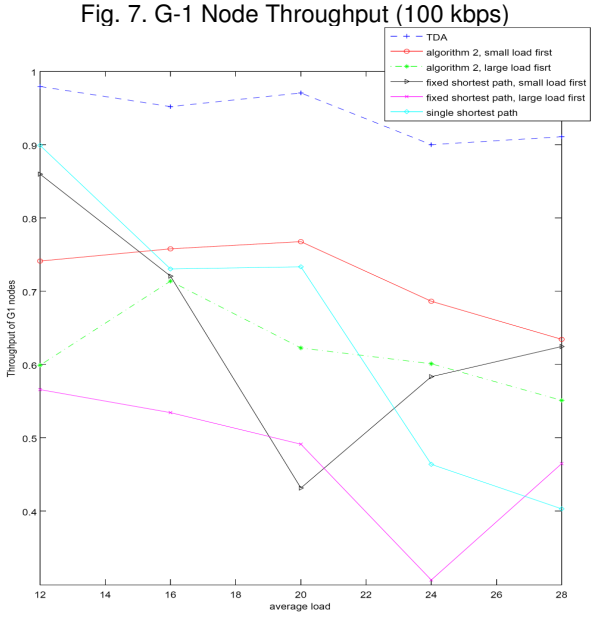

Fig. 8. G-1 Node Throughput (150 kbps)

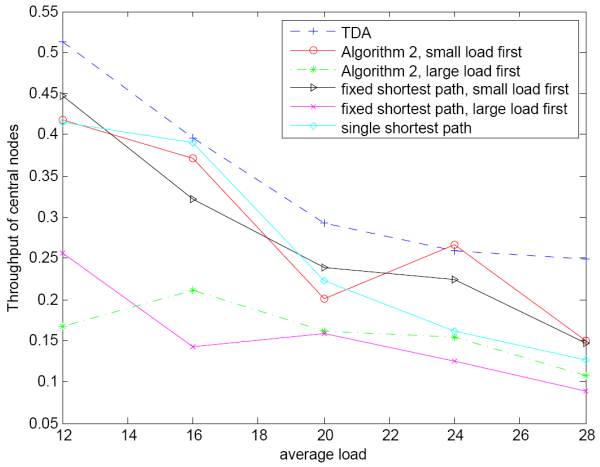

Fig. 9. Central Node Throughput (50 kbps)

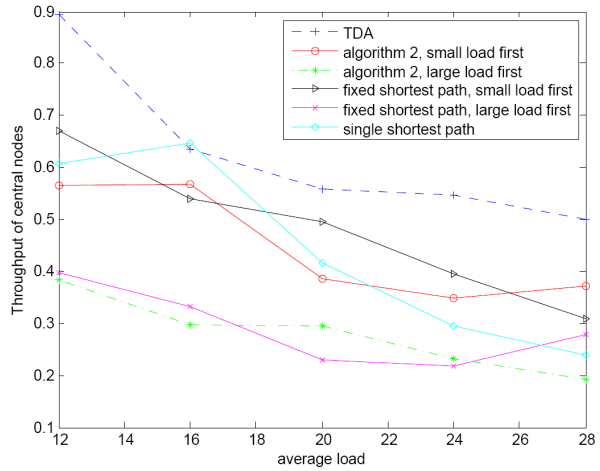

Fig. 10. Central Node Throughput (100 kbps)

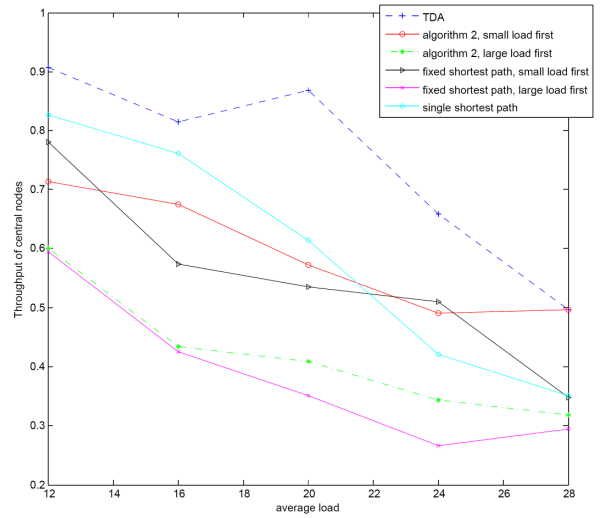

Fig. 11. Central Node Throughput (150 kbps)

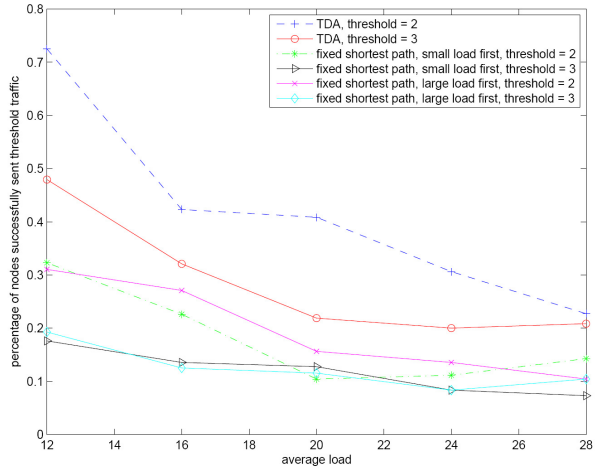

Fig. 12. Percentage Shares Received (50 kbps)

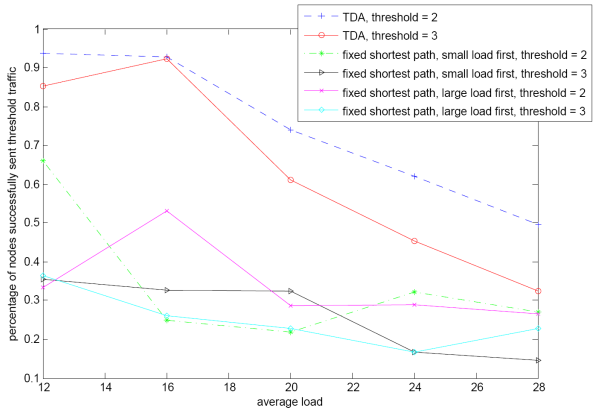

Fig. 13. Percentage Shares Received (100 kbps)

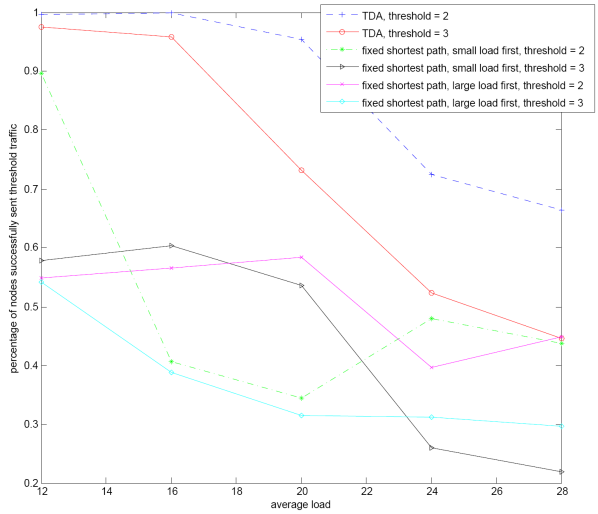

Fig. 14. Percentage Shares Received (150 kbps)

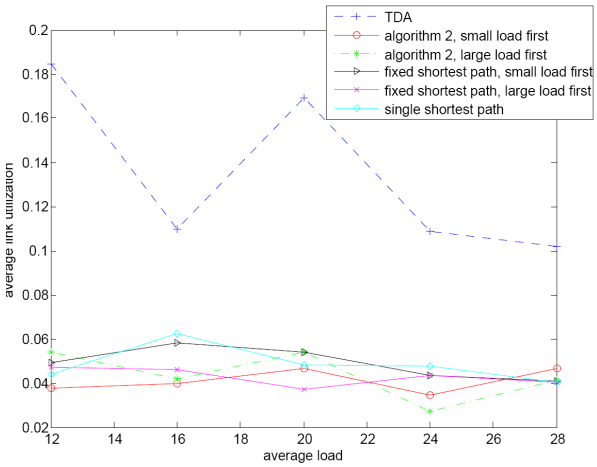

Fig. 15. Average Link Utilization (50 kbps)

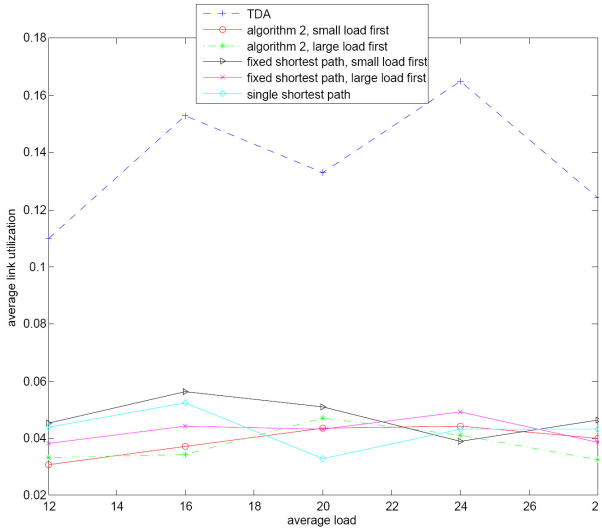

Fig. 16. Average Link Utilization (100 kbps)

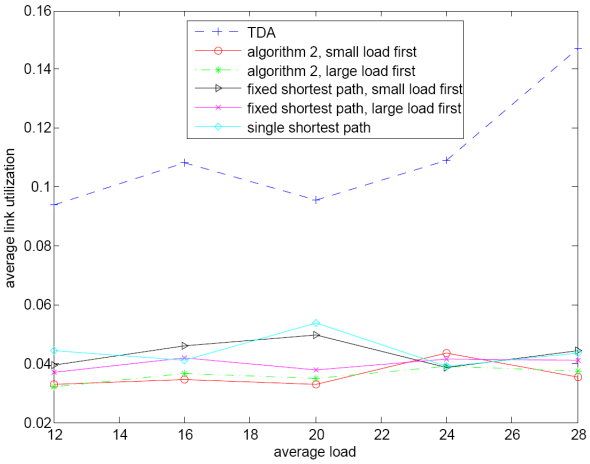

Fig. 17. Average Link Utilization (150 kbps)

7. Conclusion

This paper presents a new paradigm for load balancing in wireless mesh networks with static nodes. This is one of the first papers to explore load balancing in wireless mesh networks as a maximum throughput scheduling problem. The primary contribution of this paper is a new scheme which simultaneously achieves three goals: improved network throughput, security and resiliency, and fairness to nodes. These are important and desirable characteristics of any networking protocols. Our scheme demonstrates that by splitting the nodes' traffic across multiple gateways through intelligent traffic scheduling and node priority assignments, it is possible to substantially improve the network performance in terms of throughput, resiliency and fairness.

In this paper, we proved some basic theoretical issues, proposed several heuristics and verified the performance of our schemes through extensive simulations. The motivation for this research drives from the importance of designing efficient load balancing algorithms for improving the performance of wireless networks, and the inherent difficulty in designing such schemes. This task becomes especially challenging due to the limited availability of bandwidth in the wireless domain and the fact that obtaining maximum throughput across a network or a mesh through optimal load balancing is a known NP-hard problem.

IP routing involves packets between source and destination traversing independent paths. It is only natural to extend this traffic-splitting idea to the wireless mesh domain. Our scheme would be beneficial for real-life community wireless networks, which have a promising growth potential in the future of commercial wireless technology.

The direction of our continuing research is to integrate our scheme with existing wireless protocols. This will enable us to study the effects of other wireless network parameters like link unreliability and transmission delays. Further improvement in the dynamicity of the scheme for changing network conditions is in order. For example, stationary wireless networks with frequent node failures, or networks with strictly synchronized node sleep and wakeup cycles (due to power conservation concerns) can utilize a variant of our scheme with more dynamic route re-computations.

Our continuing research also focuses on designing a similar traffic splitting scheme for wireless mesh networks with mobile nodes. Incorporating mobility would require decisions to be made locally at the nodes. Part of our research involves theoretical analysis of the difficulty of a mobile node to establish simultaneous multiple paths versus the benefits of establishing these paths. A scheme involving simultaneous multiple paths for a mobile node could be beneficial for node throughput if some old paths survive when the node moves.

References

[1] http://www.netlib.org/utk/lsi/pcwLSI/text/node248.html

[2] O. Beaumont, V. Boudet, F. Rastello, Y. Robert, "Partitioning a square into rectangles: NP-completeness and approximation algorithms", Algorithmica, 34, 2002, pp.217-239

[3] O. Beaumont, V. Boudet, F. Rastello , Y. Robert, "Matrix multiplication on heterogeneous platforms", IEEE Trans. Parallel Distributed Systems, 12(10), 2001, pp. 1033-1051

[4] O. Beaumont, A. Legrand, F. Rastello, Y. Robert, "Dense linear algebra kernels on heterogeneous platforms: Redistribution issues", Parallel Computing, 28, 2001, pp. 155-185

[5] O. Beaumont, A. Legrand, F. Rastello, Y. Robert, "Static LU decomposition on heterogeneous platforms", Int. Journal of High Performance Computing Applications, 15(3), 2001, pp. 310-323

[6] I. Akyildiz, X. Wang, W. Wang, ``Wireless Mesh Networks: A Survey", Computer Networks Journal 47, (Elsevier), March 2005. pp. 445-487.

[7] http://wire.less.dk/wiki/index.php/MeshLinks

[8] http://www.communitywireless.org/

[9] http://research.microsoft.com/mesh/

[10] http://www.ietf.org/rfc/rfc0791.txt

[11] [KN] M. Garey, D. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", W. Freeman, ISBN 0716710455, A6: MP9, 1979, pg. 247

[12] P. Pham, S. Perreau, "Performance Analysis of Reactive Shortest Path and Multi-path Routing Mechanism with Load Balance", IEEE INFOCOM'03, San Francisco, CA, 2003.

[13] H. Hassanein, A. Zhou, "Routing with Load Balancing in Wireless Ad hoc Networks", 4th ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems, Rome, Italy, 2001, pp. 89 – 96.

[14] S. Lee, M. Gerla, "AODV-BR: Backup Routing in Ad hoc Network", IEEE WCNC'00, pp. 1311–1316.

[15] S. Roy, D. Saha, S. Bandyopadhyay, "A Network-Aware MAC and Routing Protocol for Effective Load Balancing in Ad Hoc Wireless Networks with Directional Antenna", *MobiHoc'03*, Annapolis, MD.

[16] Y. Ganjali, A. Keshavarzian, "Load Balancing in Ad Hoc Networks: Single-path Routing vs. Multi-path Routing", IEEE Infocom, Mar 2003, Hong Kong.

[17] http://pinehurst.usc.edu/~junsoo/smr/

[18] http://www.oreillynet.com/pub/a/wireless/2004/01/22/wirelessmesh.html

[19] www.mitre.org/work/tech_transfer/mobilemesh/

[20] R. Draves, J. Padhye, B. Zill, "Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks", *MobiCom'04,* Sept. 2004, Philadelphia, PA

[21] A. Raniwala, T. Chiueh, "Architecture and Algorithms for an IEEE 802.11-Based Multi-Channel Wireless Mesh Network", IEEE Infocom Mar 2005, Miami, FL.

[22] J. Hespanha, S. Bohacek, "Preliminary Results in Routing Games", American Control Conference, Arlington, VA, June 2001, vol. 3, pp. 1904-1909

[23] G. Horton, "A multi-level diffusion method for dynamic load balancing", Parallel Computing. 19 (1993), pp. 209-229

[24] D. Eppstein, "Finding the k shortest paths",35th IEEE Symposium on Foundations of Computer Science., Santa Fe, 1994, pp. 154-165.

[25] D. Ganesan, R. Govindan, S. Shenker, D. Estrin, "Highly-Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks", Mobile Computing and Communications Review, Vol. 1, Number 2, 2002.

[26] Y. Desmedt, Y. Frankel, "Threshold Cryptosystems", Proceedings on Advances in cryptology, 1989, ISBN: 0-387-97317-6 Santa Barbara, CA, pp. 307-315

[27] C. Chekuri, A. Goldberg, D. Karger, M. Levine, C. Stein, "Experimental Study of Minimum Cut Algorithms", Tech. Rep., NEC Research Institute, Inc., 1996, pp. 96-132

[28] S. Bellovin, E. Gansner, "Using Link Cuts to Attack Internet Routing", Tech. Rep., ATT Research, 2004, Work in Progress 2003 USENIX

[29] R. Gandhi, S. Khuller, A. Srinivasan, "Approximation Algorithms for Partial Covering Problems", Lecture Notes in Computer Science, Springer-Verlag GmbH, ISSN: 0302-9743, Vol. 2076, pp. 225-236