Reducing Congestion Effects in Wireless Networks by Multipath Routing

Lucian Popa[1], Costin Raiciu[2], Ion Stoica[1], David S. Rosenblum[2]
[1] Department of Computer Science
University of California, Berkeley
{popa, istoica}@cs.berkeley.edu
[2] Department of Computer Science
University College London
{c.raiciu, d.rosenblum}@cs.ucl.ac.uk

***Abstract*—We propose a solution to improve fairness and increase throughput in wireless networks with location information. Our approach consists of a multipath routing protocol, Biased Geographical Routing (BGR), and two congestion control algorithms, In-Network Packet Scatter (IPS) and End-to-End Packet Scatter (EPS), which leverage BGR to avoid the congested areas of the network. BGR achieves good performance while incurring a communication overhead of just 1 byte per data packet, and has a computational complexity similar to greedy geographic routing. IPS alleviates transient congestion by splitting traffic immediately before the congested areas. In contrast, EPS alleviates long term congestion by splitting the flow at the source, and performing rate control. EPS selects the paths dynamically, and uses a less aggressive congestion control mechanism on non-greedy paths to improve energy efficiency.**

Simulation and experimental results show that our solution achieves its objectives. Extensive ns-2 simulations show that our solution improves both fairness and throughput as compared to single path greedy routing. Our solution reduces the variance of throughput across all flows by 35%, reduction which is mainly achieved by increasing throughput of long-range flows with around 70%. Furthermore, overall network throughput increases by approximately 10%. Experimental results on a 50-node testbed are consistent with our simulation results, suggesting that BGR is effective in practice.

I. INTRODUCTION

Wireless embedded processors contained in mobile phones, handheld devices or weaved into the environment as sensors, are likely to become the main part of the future Internet [9]. Furthermore, it is expected that location information will be widely available for such processing, to enhance context-aware types of interactions [9].

The prospect of having ad-hoc wireless networks composed of numerous location-aware nodes spread in the surrounding environment (such as SmartDust [1]) poses new interesting challenges to the research community. Congestion in wireless networks has already been explored by other research, observing its impact on performance: a drastic decrease in throughput [28] and increased per-packet energy consumption [11]. On the other hand, computing is moving to an era where applications require large and stable bandwidths to perform their tasks. Such applications include multimedia applications, high frequency sensing applications, file transfer, and so forth. If devices enabling these applications are going to become an integral part of tomorrow's networks, solutions to reduce the effects of congestion in wireless networks are required.

A promising approach for routing in such networks is geographical routing, an algorithm that leverages location information to route messages in a hop-by-hop, greedy manner. Assuming that a coordinate system is in place (either GPS or other coordinate systems, such as NoGeo [2], BVR [4] or [3]), this scheme is scalable, has low computational overhead and requires minimum routing information to be maintained by nodes.

However, shortest path routing schemes in general, and geographical routing in particular, amplify the effects of congestion: in a random communication pattern, the nodes in the center of the network carry a disproportionately large amount of the entire traffic, drastically decreasing the throughput of the flows they forward. This affects most long-range flows, as they have a higher probability of intersecting the central hotspot.

In this paper, we present a solution that seeks to utilize idle or under-loaded nodes to reduce the effects of congestion. To achieve this goal, we enhance geographic routing to allow a source to select different paths towards the destination. While multi-path solutions for geographic routing have been proposed before, they have either limited effectiveness (e.g., waypoint routing), or they exhibit a high overhead (e.g., TBF [5]). At this end, we propose Biased Geographical Routing (BGR), a lightweight, stateless, geographical forwarding algorithm, as a cost-effective complement to greedy routing. BGR routes packets on curved trajectories, by forwarding packets along curves, instead of along the shortest path, towards the destination.

To further mitigate congestion, we design two congestion control mechanisms that leverage BGR:

- *In-Network Packet Scatter (IPS)* is a lightweight mechanism that aims to relieve transient congestion by locally splitting the traffic along multiple paths to avoid congested hotspots.
- *End-to-End Packet Scatter (EPS)* is an end-to-end mechanism that aims to alleviate longer term congestion, when IPS fails. EPS works by splitting the flow at the source, and performing independent rate control along each path in response to congestion.

We have evaluated the performance of BGR by using a high-level simulator, a packet-level simulator (ns2 [6]), and a testbed comprising 90 nodes [21]. The results show that BGR is a practical and efficient multipath routing algorithm. We have evaluated IPS and EPS using ns2. Simulation results show that their combined action:

- increases network throughput for long flows with around 70% when compared to greedy routing,
- increases fairness by reducing the dependence of flow throughput on the distance between the endpoints, and
- increases overall network throughput by around 10%.

In addition, we have evaluated the potential of multipath routing to increase the network throughput on the Mirage testbed [21]. Experimental results are consistent with the simula-

1-4244-0593-9/06/$20.00 ©2006 IEEE

tion results, showing that multipath routing is a viable solution for increasing throughput.

This paper is structured as follows. Section II presents the BGR algorithm and a high-level simulation analysis of its performance. In Section III, we show how multipath routing using BGR can be used to increase throughput and fairness among flows with different lengths. In Section IV, we evaluate IPS and EPS through ns2 simulation. Section V discusses the deployment of BGR in TinyOS and the results obtained from a testbed deployment. Section VI provides an overview of related work. Finally, conclusions and future directions are discussed in Section VII.

II. Biased Geographical Routing (BGR)

In this section, we describe the requirements of our solution and the details of the BGR algorithm. In addition, we present simulation results that show that BGR achieves good performance with a low overhead.

A. Design goals

We design our solution to work in any wireless network with coordinate based routing. To accommodate sensor networks, we require our solution to work under stringent energy and computational constraints, which characterize these networks.

Next, we summarize the requirements of the geographic routing protocol to be used by our solution:

- *Low communication overhead* – typically, packets sent by the sensor nodes are very small (e.g., in TinyOS the maximum packet size is 29 bytes), emphasizing the requirement for low communication overhead.
- *Simplicity* – the routing algorithm must have *low computational overhead* – to allow timely execution on slow processors and to minimize energy usage – and *low memory footprint* to fit into memory (e.g., the micaz mote has only 4 kB of RAM).
- *Low state* – nodes must maintain a minimal amount of state, to allow the network to scale up to a large number of devices (i.e. no per-flow or per-path state in network)

In addition, to effectively avoid the hotspots in the network, the multi-path algorithm should be able to provide a large number of paths with few common hops (this will be referred to as *path overlap*) without increasing routing failures, as compared to the single-path greedy routing.

B. BGR Description

The main idea behind our solution is to insert a "*bias*" in each packet, which determines the curvature of the path followed by the packet towards the destination. The bias is a measure of how far the trajectory will deviate from the greedy route and also indicates the side of the deviation. In our implementation, the bias is treated at each hop as an angle. Instead of routing greedily towards the destination D, BGR routes greedily towards the point N_2 (target point) situated at a predefined distance from the current node N_1 such that the angle between the lines N_1N_2 and N_1D is equal to the bias. In this way, initially, packets with different biases are scattered on different trajectories and later start getting closer and closer to the destination.

To avoid spiral trajectories, we decrease the modulus of the bias at each hop with a value inversely proportional to the square of the distance to the destination from the current node

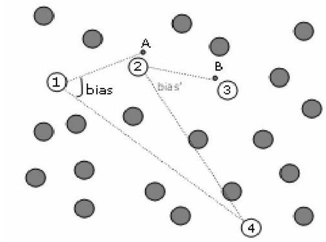


Figure 1. BGR Forwarding

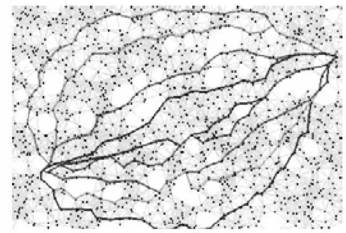


Figure 2. BGR in the simulator

(we borrowed the idea from physics as a parallel to natural forces like gravity): $bias = bias - K/d^2$. When the modulus of the *bias* reaches 0, the *bias* is not modified any longer, to avoid "missing" the destination; from this point on, the packet is routed greedily towards the destination. The proportionality parameter K depends on the size of the network, the average number of neighbors and the radio range of the nodes. However, the algorithm is quite resilient to this parameter: in our tests, we achieved similar results for a large range of values (see subsection F).

Fig. 1 illustrates the BGR forwarding process. Node 1 holds a packet with bias value *bias* that it must route towards the destination, node 4. Node 1 selects target point A on the biased line at a predefined distance (usually the maximum radio distance) and selects the next hop for the packet to be one of its neighbors that is closest to A, in this case node 2. Before forwarding the packet to node 2, node 1 decreases the initial bias value to *bias'* and sends the packet to node 2. In turn, node 2 will select node 3 as the next hop and further decrease the bias. This process continues until the packet reaches the destination or the packet is dropped, if no neighbor is closer to the target point than the current node.

Fig. 2 shows a sample trace of the protocol in our high level Java simulator. In the simulation, a single source sends packets with different biases towards the same destination. The bold lines represent the paths of the packets, while gray lines represent links between neighboring nodes.

C. Properties of BGR and other considerations

BGR is both *simple* and has *low overhead*. The computational overhead is close to greedy routing. BGR adds two simple operations to greedy routing: (1) plot the virtual destination point, and (2) decrease the bias, both of which are independent of the number of neighbors of a node. The communication overhead consists of adding the bias to the packet header. The bias can be encoded in as little as one byte. One byte allows an integer representation of the angle between -127° and 127°, which in our experience is enough in practice.

The setup cost and memory usage of the protocol are the same as that of greedy routing, since all the protocol requires is knowledge of neighbors and their locations.

Next, we give a simple convergence property of BGR.

Convergence Property. BGR avoids infinite loops.

Proof. If $K>0$, the bias will eventually reach zero since at each step it is decreased with a positive value Δ, where $\Delta > K/D^2$, D being the diameter of the network. Thus, BGR does not degenerate in routing loops.

If we only use integer values for *bias*, we need to make sure that the bias is decreased at all steps by at least 1. In the most general case we need to set $K \geq D^2$ to ensure that BGR converges. ■

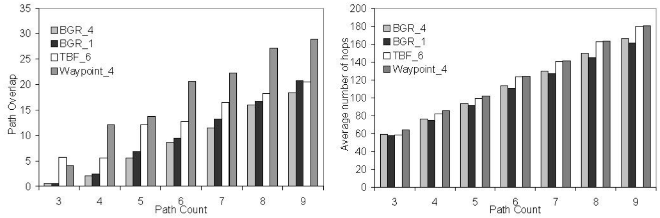


Figure 3. Path distinctiveness and cost

A particular case that appears often in practice is when $|bias| < \pi/2$. In this case, BGR converges for much smaller values of K. Actually, BGR converges even for $K=0$ by forming spiral trajectories, if we assume perfect next hop selection (i.e., at each hop, BGR's target point always coincides with one of its neighbors' position). In practice, for non-pathological cases (e.g. a circle of close-by nodes of diameter D and the destination node in the center), BGR converges regardless of the value of K.

Fallback Mode. A usual approach in geographic routing protocols is to enter a fallback mode when greedy forwarding is impossible (see GPSR [15], CLDP[16]). This solution is orthogonal to our algorithm; BGR can adopt the same policy by preserving or resetting the bias upon recovery from fallback mode. Our current implementation does not support fallback mode, for two reasons: first, we wanted to prevent any particular algorithm from influencing our comparisons; and second, we focused on dense networks, where greedy routing alone achieves high availability.

Coordinate Systems. BGR requires coordinate information to function properly, and GPS should be used when available. However, the algorithm also works on virtual coordinate systems with Cartesian properties, such as No-GEO [2] or DV-hop[3]. Tests we performed using the No-GEO coordinate system showed that, without any modifications, BGR functioned properly.

BGR also requires a location service that allows a node to find the location of another node, given a node identifier. Any of the location services proposed in the literature can be used, for instance the Grid Location Service [26].

Other choices. We have considered other routing schemes before settling for BGR. In one scheme, the packet is assimilated to a particle that has an initial speed and is attracted towards the destination. Another approach was to plot all nodes on a sphere with the poles in the two communication endpoints and route packets on longitudes. Simulation results showed that these approaches have higher overhead and worse performance (i.e.higher path overlap) when compared to BGR.

D. *Applications of BGR*

BGR's requirements target sensor networks as these represent the most resource-scarce wireless environments; we stress that BGR's use is not limited to sensor networks, the algorithm being applicable to any wireless network.

The list of possible applications for BGR includes multipath routing and all its possible uses. Multipath routing can be easily implemented by sending packets with different biases towards the destination. Energy fairness can be increased by sending messages on biased trajectories, to avoid the nodes in the center of the network. Fault-tolerant routing can be achieved by replicating packets on different trajectories towards the destination.

For example, mesh networks [24], foreseen as a future way of providing Internet, could become large enough to benefit greedy routing. The advantages of multipath routing are immediate: users could use BGR to avoid sending traffic on the greedy route towards the network uplinks, to avoid congested or even faulty nodes.

In this paper we only focus on a single application of BGR, alleviating congestion in dense wireless networks.

E. *BGR Comparative Analysis*

We present high-level simulation comparisons of BGR with our implementations of TBF [5] and waypoint routing. The purpose of the comparison is to evaluate how BGR's low overhead affects its performance. We use as performance metrics path overlap (measured as the fraction of common hops shared by different paths) and communication overhead (measured in bytes).

Simulation Setup. We coded and used a high-level Java simulator that offers scalability, ease of implementation and high testing speed. In subsequent sections we evaluate BGR using ns2 and a real deployment.

Tests were run on a network of 3300 nodes randomly placed in a 500x500m geographic area. The radio range is 20m resulting in each node having close to 16 neighbors on average. Given our target metrics, the simplicity of the scenario does not affect the comparison. Measurements were made for multiple random deployments and averaged over a few thousand experiments. In our tests, the ratio between the longest and shortest paths is around 1.4 (the length of a path is its hop count).

In our implementation, TBF uses circular trajectories. Instead of embedding the parametric equation of the circle in each packet, we reduce communication overhead by embedding only the coordinates of the center of the circle along with a bit identifying a "left" or "right" trajectory and the current and the final steps on the parameterized trajectory. We use for testing a version of TBF that is several times more computationally intensive than BGR: at each forwarding hop, TBF computes on average 9 points on the trajectory and checks their proximity to the neighbors. We believe this instance of TBF is representative in terms of performance and overhead for all TBF variants usable in practice. However, these tests are by no means exhaustive over all possible instances of the two algorithms.

Results. Fig. 3 shows a comparison of path overlap for BGR, TBF and a simple implementation of waypoint routing using a single routing landmark. BGR's parameter K is set to 500. The names of the data series include the name of the algorithm and the per packet communication overhead, in bytes. We considered locations encoded on four bytes (two for each dimension). BGR_4 encodes the bias as a float (4 bytes); BGR_1 encodes the bias (in degrees) as a single byte. Implementations were tuned such that BGR used a lower number of hops overall (on all paths) as shown in the second chart; this ensured that the rest of the algorithms used wider paths that are less prone to overlapping.

The results show that, at similar costs, both instances of BGR achieve lower path overlap as compared with TBF and waypoint routing. In theory, TBF can approximate any trajectory (including BGR trajectories) and thus can obtain minimal path overlap. The main point of the comparison is not that

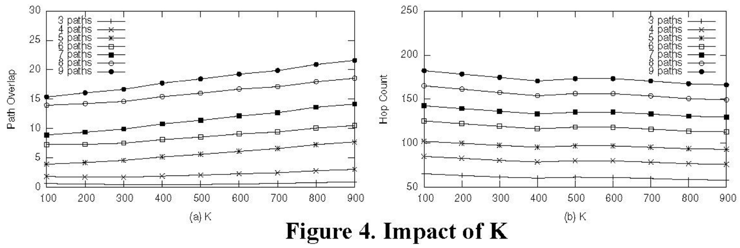


Figure 4. Impact of K

BGR performs better but that significant simplifications do not degrade the desired performance metrics.

F. The influence of parameter K

BGR has similar behavior for a large range of values for *K* and, thus, the method of selecting *K* (e.g. through sampling) should not be particularly important. Fig. 4 plots the path overlap for different values of the parameter *K* and different number of paths used (using the same high level simulator as before), as well as the average number of hops to estimate costs particular to multipath routing. As we can see, when *K* varies in the range 100 to 800, the variation of path overlap is small – for instance, the variation for "4 paths" is only 1%. As expected, when using a lower value for *K* paths are wider, having lower overlap. However, longer paths also imply higher costs and higher probability to encounter voids. Selecting *K* needs also to take into account the end-purpose of multipath routing; for instance, a larger value could be selected if BGR is used to avoid local obstacles on the way, such that paths converge quickly to greedy (we use this approach with IPS). Choosing *K* on a per flow basis as a function of the distance between the endpoints is another possibility that we intend to explore in future work.

III. Mitigating Congestion in Wireless Networks

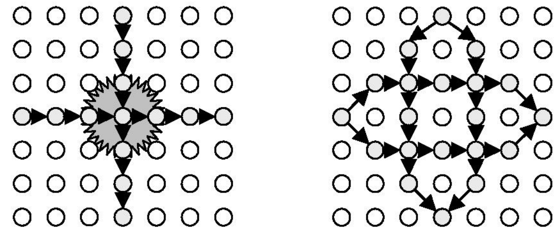
Congestion collapse in wireless networks has particularities such as spatial correlation that cause even idle nodes to become congested when the wireless area around them is busy. Unlike the Internet where congestion is mostly situated at the border of the network, in wireless networks with point-to-point communication congestion usually builds in the center. Fortunately, the connectivity of most wireless networks is rich enough to allow routing packets on alternate paths that avoid the congested areas.

A. Traffic Assumptions

We assume a point-to-point communication pattern with randomly chosen endpoints where devices operate at a low duty cycle and become suddenly active in response to context changes. Examples include the detection of interesting events in sensor networks or the appearance of information of interest (e.g., an on-site multimedia service) in a mobile node's vicinity. In such cases, devices send packets at high rates towards the attraction points scattered throughout the network.

Our traffic model consists of independent packets, where the benefit of the receiver increases linearly with the number of packets received, regardless of their ordering. The latter assumption is important, as our solution changes the ordering of the packets through multipath routing. File transfer applications and sensor readings adhere to this model. Reliable delivery can be implemented were needed, on top of the mechanisms we describe, by using techniques such as forward error correction and selective acknowledgements.

In our solution, we assume that there are areas close to the hotspots that are not congested and can be used to reroute the traffic towards the destination. In our communication model, traffic is usually generated in bursts coming from isolated nodes or small areas of the network, and thus, most of the network is not heavily loaded.



(a) **(b)**

Figure 5. Increasing throughput with multipath routing

B. Solution Outline

The underlying idea of our solution is straightforward. When a flow experiences congestion, we split its traffic onto multiple paths, in an attempt to "spread" network load on a wider area and thus alleviate congestion. In this way, the newly created flows will carry packets at a slower rate, allowing the hotspot in the network to be relieved. A simple example of this approach is presented in Fig. 4(a) where two flows carrying packets at a high rate (assume the rates equal R) create a hotspot around the intersection area (assuming the wireless capacity is less than 2R). In Fig. 4(b), the flows are split into two non-interfering paths; at each intersection area data will flow at aggregate rate R, alleviating congestion.

The underlying assumption of our proposal is that the interference among the split paths is negligible; obviously, the paths will be interfering close to the endpoints, but we require the flow intersection points don't interfere. This is a reasonable assumption if we use paths that are split far enough apart and the distance from the source to the destination is (significantly) larger than the contention range. To enforce this behavior, we use a lower bound on the geographical distance between the endpoints to decide whether a flow should be split.

Our solution has two components, aiming to tackle different conditions in the network: *in-network packet scatter* (IPS) and *end-to-end packet scatter* (EPS). IPS scatters packets close to congested areas, attempting to deal with transient congestion in a fast way. EPS reroutes the flow starting from the source along a small number of paths, when detecting congestion, and reduces the sending rate if congestion persists. EPS aims to deal with persistent congestion and is suited for long-lived flows. We will now discuss both mechanisms in more detail.

C. IPS - In-Network Packet Scatter

IPS splits flows close to the congestion point. Each node monitors the congested status of all its neighbors and splits the flows that are going towards a congested neighbor, if the node itself is not congested. The scattered packets contain large biases, such that the modified trajectories quickly move away from the original trajectory. However, to counteract the negative effect of inserting large biases (i.e. creating long paths), we use a modified version of BGR that has a larger value for *K*, ensuring that the bias is quickly decreased and that the routes do not deviate too much from the greedy route. We used a value for *K* that is 100 times larger than our usual *K*; however, we obtained similar results for a wide range of values. To indicate the use of one of the two values for K we added one extra bit to each packet.

When forwarding traffic towards a congested neighbor, packets will be individually scattered with uniform probability by modifying the bias with one of several preset values (in our evaluation we use three values $-\pi/2$, 0, $\pi/2$). This simple, uniform split contributes to the efficiency of IPS, essentially a lightweight solution that does not maintain per flow information and deals with short-term, transient, congestion. IPS responds quickly to network conditions and has small energy consumption (because the multiple paths are created closer to the destination). Moreover, it only requires nodes to maintain information about the congested state of their neighbors, without maintaining per flow information. The pseudocode for the IPS algorithm is presented in Fig. 6.

1) Congestion Detection

To detect congestion, we rely heavily on previous work ([11], [12]). Our detection mechanism is based on buffer occupancy and wireless usage, exponentially averaged to eliminate noise. Wireless usage is measured by periodically sampling wireless medium. We also consider that the medium is busy when the node's MAC is in backoff mode. In this way, nodes that are busy (i.e. have packets to forward) will detect wireless usage accurately.

2) Congestion Signaling

IPS requires constant information exchange with neighbors, incurring additional overhead. To minimize this overhead, our implementation is similar to FUSION [12], where a single bit indicating congestion is added to each packet and each node promiscuously listens to the packets sent by its neighbors to detect their congested status.

D. EPS - End-to-End Multipath Packet Scatter

If IPS cannot successfully support the aggregate traffic (i.e. avoid congestion), it will only scatter packets to a wider area potentially amplifying the effects of congestion collapse due to its longer paths (a larger number of contending nodes lead to a larger probability of loss). In such cases a closed loop mechanism is required to regulate the source rates. EPS is applied at the endpoints of the flows, and regulates the number of paths the flow is scattered on and the rate corresponding to each path. The source requires constant feedback from the destination regarding network conditions, making this mechanism more expensive than its local counterpart.

The idea behind EPS is to dynamically search and use free resources available in the network in order to avoid congestion. When the greedy path becomes congested, EPS starts sending packets on two additional side paths obtained with BGR, searching for free resources. To avoid disrupting other flows, the side paths perform more aggressive multiplicative rate decrease when congested.

EPS dynamically adjusts to changing conditions and selects the best paths to send the packets without causing oscillations. The way we achieve this is by doing independent congestion control on each path. If the total available throughput on the three paths is larger than the sender's packet rate, the shortest path is preferred (this means that edge paths will send at a rate smaller than their capacity). On the other hand, if the shortest path and one of the side paths are congested but one other side path has unused capacity, our algorithm will naturally send almost all the traffic on the latter path to increase throughput. The EPS algorithm detects congestion in the same way as IPS. Pseudocode for a simplified version of EPS in presented in Fig. 7.

```
forwardPacket (Node crt, Packet p) {
  Node next = chooseBGRNextHop(p);
  If(next.isCongested() && !crt.isCongested && !p.IPSsplit){
     p.IPSsplit = true; // sets IPS bit
     choice = random_uniform {Bias Set}  //e.g. {-α,0,α};
     p.setBias(p.bias+choice);
     next = chooseBGRNextHop(p);
  }
  sendLinkLayerPacket(next,p);
}
```

Figure 6. IPS Algorithm (pseudocode)

```
//For simplicity, we assume a single destination and three paths
MaxPaths = 3; bias={ 0, 45°,-45°}; reduce_rate= {0.85, 0.7, 0.7};
//sender side pseudocode
receiveFeedback (int path, bool flowCongested) {
  if (!EPS_Split) //not already split
       if(flowCongested) splitSinglePath();
       else sendingRates[0] += increase_rate; //additive increase
  else //we have already split the flow into multiple paths
      if(flowCongested) sendingRates[path] *= reduce_rate[path];
      else { // no congestion, we increase the path sending rate
          if(path == 0) { // main path
             sendingRates[0] += increase_rate; //additive increase
             totalAvailableRate = sum(sendingRates);
             if(totalAvailRate > 1) {//we can transmit more than we want
                diff = 1 – totalAvailableRate;
                for(int i = 1; i < MaxPaths; i++)
                   sendingRates[i] – = diff*sendingRates[i]/
                                  ( totalAvailableRate - sendingRates[0]);
             }
          }
          else sendingRates[path] += min(increase_rate,
                                          1-sum(sendingRates))
      }
}
splitSinglePath(){
   for(int i = 0; i < MaxPaths; i++) sendingRates[i] = 1 / MaxPaths;
   EPS_Split = true;
}
sendPacketTimerFired(){
  path_choice = LotteryScheduling(sendingRates);
  Packet p = Buffer.getNext();  //orthogonal buffer policy
  p.split = EPS_Split;  // if we split or not
  p.bias = bias[path_choice];
  next = chooseBGRNextHop(p);
  …//other variables
  sendLinkLayerPacket(next,p);
}
// receiver side pseudocode
receivePacket(Packet p){
  receivedPackets[p.source][p.path]++;
  if(p.congested)   congestedPackets[p.source][p.path]++;
  if(receivedPackets[p.source][p.path] > messagesPerAck) {
     boolean  isCongested = congestedPackets[p.source][p.path]  >
                                        packets[p.source][p.path]/2);
     sendFeedback(p.source, isCongested);
      …//reinitialize state variables
  }
}
```

Figure 7. EPS Algorithm (pseudocode)

1) Congestion Signaling

Choosing an appropriate closed loop feedback mechanism impacts the performance of EPS. Unlike WTCP [18] which monitors packet inter-arrival times or CODA [11] which does local congestion measurements at the destination, we use a more accurate yet lightweight mechanism, similar to Explicit Congestion Notification (ECN) [13]. Nodes set a congestion bit in each packet they forward when congestion is detected. In our implementation, the receiver sends state messages to the sender to indicate the state of the flow. State messages are triggered by the receipt of a predefined number of messages, as in CODA.

The number of packets acknowledged by one feedback message is a parameter of the algorithm, which creates a tradeoff between high overhead and accurate congestion signaling (e.g., each packet is acknowledged) and less expensive but also less accurate signaling.

The destination maintains two counters for each path of each incoming flow: *packets* counts the number of packets received on the path, while *congested* counts the number of packets that have been lost or received and have the congested bit set to 1. When *packets* reaches a threshold value (given by a parameter called *messages_per_ack*), the destination creates a feedback message and sends it to the source. The feedback is negative if at least half of the packets received by the destination have the congestion bit set, or positive otherwise. As suggested in the ECN paper [13], this effectively implements a low pass filter to avoid signaling transient congestions, and has the positive effect that congestion will not be signaled if it can be quickly relieved with our IPS.

2) RTT estimation

When the sender starts the flow, it starts a timer equal to:

$$messages\_per\_ack \,/\, packet\_rate + 2 \cdot hopcount \cdot hop\_time.$$

We estimate *hop_count* using the expected inter-node distance; *hop_time* is chosen as an upper bound for the time taken by a packet to travel one hop. Timer expiration is treated as negative feedback. A more accurate timer might be implemented by embedding timestamps in the packets (such as WTCP, TCP) but we avoid that due to energy efficiency considerations. However, most times the ECN mechanism should trigger the end-to-end mechanism, limiting the use of timeouts to the cases when acknowledgements are lost.

3) Rate control

When congestion persists even after the flow has been split at the source, we use congestion control (AIMD) on each individual path to alleviate congestion. When negative feedback is received, multiplicative decrease is performed on the corresponding path's rate. We use differentiated multiplicative decrease that is more aggressive on exterior paths than on the greedy path, to increase energy efficiency; effectively, this prioritizes greedy traffic when competing with split traffic. Additive increase is uniform for all paths; when the aggregate rate of the paths exceeds the maximum rate, we favor the greedy path to increase energy efficiency. More specifically, if the additive increase is on the shortest (central) path, exterior paths are penalized proportionally to their sending rate; otherwise, the rate of side path is increased only up to the overall desired rate (see pseudocode in Fig.7).

E. Discussion

As opposed to IPS, EPS is suited for long lived flows and adapts to a wider range of traffic characteristics, relieving persistent or wide-spread congestion when it appears. The paths created by this technique are more symmetric and thus further away from each other, resulting in less-interference. The mechanism requires each end-node maintain state information for its incoming and outgoing flows of packets, including number of paths, as well as spread angle and send rate for each path. The price of source splitting is represented by the periodic signaling messages. If reliable message transfer is required, this cost is amortized as congestion information can be piggybacked in the acknowledgement messages.

F. Limitations

When congestion is widespread and long-lived, splitting might make things worse since paths are longer and the entire network is already congested. However, as we show in the Evaluation section, this only happens when the individual flow throughput gets dramatically small (10% of the normal value) and when the costs of path splitting – in terms of loss in throughput – are insignificant.

Also, if paths interfere severely, splitting traffic might make things worse due to media access collisions, as more nodes are transmitting. This is not to say that we can only use completely non-interfering paths. In fact, as we show in Section V, our approach exploits the tradeoff between contention (when nodes hear each other and contend for media) and interference (nodes do not hear each other but their packets collide): throughput is more affected by high contention than by interference.

IV. Evaluation of IPS and EPS

In this section we present simulation results obtained through ns2 simulations [6]. We use two main metrics for our measurements: throughput increase and fairness among flows.

We ran tests on a network of 400 nodes, distributed uniformly on a grid in a square area of 6000m x 6000m. We assume events occur uniformly at random in the geographical area; the node closest to the event triggers a communication burst to a uniformly selected destination. To emulate this model we select a set of random source-destination pairs and run 20-second synchronous communications among all pairs. The data we present is averaged over hundreds of such iterations. The parameters are summarized in Table 1.

An important parameter of our solution is the number of paths a flow should be split into and their corresponding biases. Simulation measurements show that the number of non-interfering paths between a source and a destination is usually quite small (more paths would only make sense on very large networks). Therefore we choose to split a flow exactly once into 3 sub-flows if congestion is detected. We prefer this to splitting in two flows for energy efficiency considerations (the cheaper, greedy path is also used). We have experimentally chosen the biases to be +/-45 degrees for EPS and +/- 90 degrees for IPS.

TABLE 1. SUMMARY OF PARAMETERS

Parameter	Value	Parameter	Value
Number of nodes	400	Link Layer Transmission Rate	2Mbps
Area size	6000m x 6000m	RTS/CTS	No
MAC	802.11	Retransmission count (ARQ)	4
Radio Range	250m	Interface queue	4
Contention Range	550m	Packet size	100B
Average Node Degree	8	Packet frequency	80/s

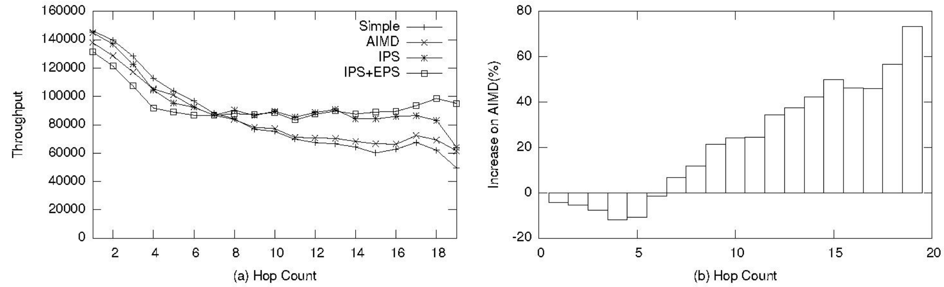


Figure 8. Throughput vs. Distance

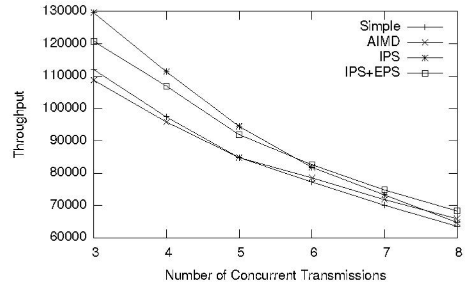


Figure 9 Throughput vs. Transmissions

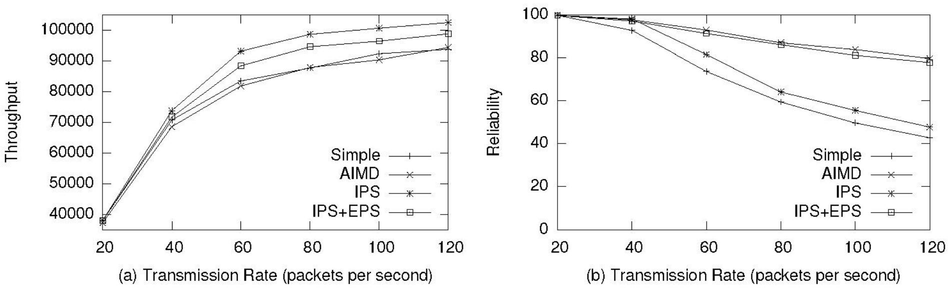


Figure 11. Impact of Source Rate

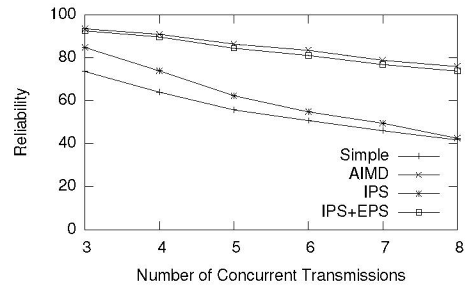


Figure 10 Received vs Transmissions

We select multiplicative decrease cutoff values through simulations, aiming to maximize throughput. We use a value of 0.85 for the greedy path (in both our algorithm and the single path AIMD) and 0.7 for the side paths. We analyzed different cutoff values for side paths and found that throughput only varies with 3% when the non-greedy paths' cutoff factor varies in the range [0.3-0.85]; 0.7 achieves close to maximum throughput and is more energy efficient than 0.85.

A. Results Summary

As expected, our solution works well for flows where the distance between the source and the destination is large enough to allow the use of non-interfering multiple paths. The EPS + IPS combination increases long-range flow throughputs with around 70% as compared to single path transmission (both with and without AIMD). For short-range flows, where multiple paths cannot be used, the throughput obtained by our solution is smaller with at most 14%, as the short-range flows interfere with split flows of long-range communications. However, by increasing long-range flows' throughput we improve fairness among the different flows achieving a lower throughput variance across flows with different lengths by 35% compared to a single path with AIMD. Moreover, the overall throughput is increased with around 10% for a moderate level of load (e.g. 3-6 concurrent transmissions).

Finally, we show that our algorithm (IPS+EPS) does not increase the number of losses compared to AIMD.

B. Throughput Variation with Distance

Fig. 8.a plots throughput as a function of the hop distance between the source and destination nodes. Single path greedy routing is shown as "Simple" while "AIMD" represents single path with rate control. The results are for 5 concurrent transmissions. The relative throughput increase of IPS+EPS compared to AIMD is presented in Fig. 8.b.

For long distance flows, the combination IPS+EPS achieves up to over 70% increase as compared to single path routing (both with and without AIMD).

In fact, our algorithm works increasingly better as the network diameter increases (preserving the ratio of transmissions per node). Given the large contention range (550m), we chose to split from the source only long-range flows – flows where the distance between endpoints is larger than 1200m (6-8 hops). Shorter flows (less than 3 hops) are affected by exterior paths of split long-range flows. Short to medium range flows are more affected; 4-5 hops are most affected having a 15% throughput drop.

IPS alone incurs smaller throughput penalties for short-range flows, but is less beneficial to long-range flows.

We observe that the throughput of single path transmissions gets lower as the distance increases, as longer paths have higher probability of intersecting other flows and hence of losing packets. On the other hand, the throughput for our solution (IPS + EPS) remains almost the same as the distance increases. This translates into improved fairness between flows. We computed the variation of the throughput for all the flows in our measurements and found that IPS+EPS's variance is 36% lower when compared to "AIMD" and twice as small as the variance of "Simple".

Finally, network throughput is increased by a little over 7% when using IPS+EPS and by 8% when using only IPS. This represents the combined increase for both the short and the long distance flows and will increase for larger networks and decrease for smaller networks.

Note that this chart looks similar when drawn for a different number of transmissions or for a different rate.

C. Impact of number of transmissions and rate

Fig. 9 presents how the number of transmissions in the network affects the average flow throughput. Throughput drastically decreases as the network becomes congested regardless of the mechanism used. For moderate number of transmissions (3-5) the combination IPS+EPS increases the overall throughput by around 10%. IPS achieves the largest throughput. However, it is not using rate control and a lot of the sent packets are lost, leading to inefficiency (see Fig. 10).

Fig. 10 shows that the combination IPS+EPS has a similar packet loss rate to "AIMD". Fig. 11.b shows this is also true

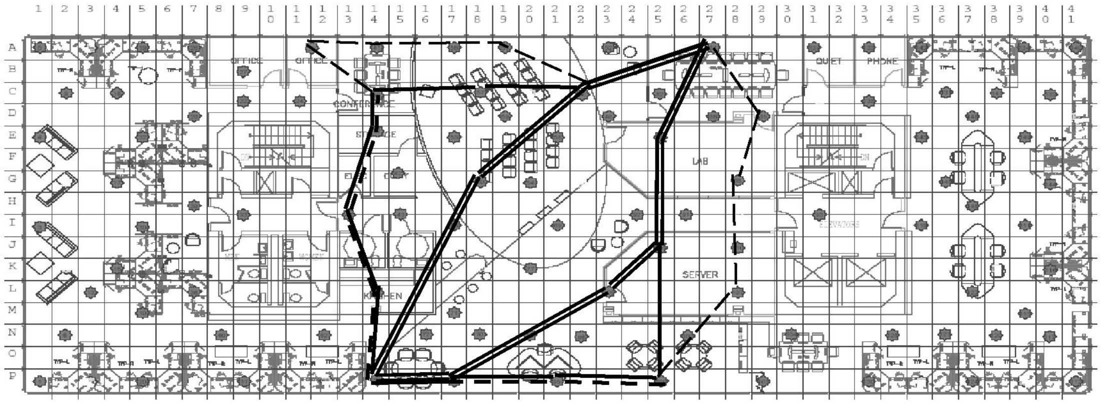


Figure 12. Paths from node A:27 to node P:14 on the Mirage Intel Lab Testbed

– – – – 50° bias ——— 40° bias ═══ 25° bias Greedy=lower 25° bias trajectory

when the transmission rate varies. This is important on two counts: first, for energy efficiency reasons, and second, to implement reliable transmission.

Fig. 11.a displays the overall throughput for different transmission rates. As we can see the throughput flattens out as congestion builds in the network but the (small) overall increase remains approximately steady.

V. IMPLEMENTATION IN TINYOS

To prove our claims, we have implemented BGR and multipath routing on the largest wireless testbed we had available, the Berkeley Intel Research Lab testbed called Mirage [21]. This sensor network testbed has 95 MicaZ [22] motes. The topology of the testbed is presented in Fig.12.

We implemented BGR in TinyOS [19], the de-facto standard development environment for sensor networks. We tested our implementation in the TOSSIM [20] simulator, and on the testbed. Here we present the results from the testbed. The results suggest that BGR can be used in real life and that sending packets on multiple paths has the potential of obtaining higher throughputs and of mitigating congestion.

A. Notes on the implementation

To implement BGR in TinyOS we had to implement the trigonometric functions of *atan*, *sin* and *cos*. We use simple and accurate approximation algorithms, all contained within 30 lines of C code. As compared to greedy routing, the BGR algorithm adds just one call for each of the above functions. The extra overhead is negligible in regards to the other computational tasks motes have to accomplish. To select neighbors, we used periodic broadcasting and an exponential average of the link quality estimation (LQI) provided automatically by the CC2420 radio [23].

B. Experimental Setup

In order to increase path lengths, we reduced the radio transmission power of the motes to the smallest power that kept the testbed reasonably connected (-22 dBm). Even so, nodes from one side of the network can occasionally receive messages from nodes on the opposite side (vertical in Fig. 12). Thus, our primary assumption of non-interfering paths is not true on this testbed. Consequently, we used for testing only the central part of the testbed formed by a rectangular continuous mesh of 45 nodes.

In this context, the main result of our experiment is the tradeoff between contention (nodes hear each other and refrain from sending) and interference (nodes do not hear each other but packets are lost due simultaneous transmissions). High rate intersecting greedy paths create contention in the center of the network that leads to congestion; in this case sending data on the edges of the network is desirable. At lower transmission rates, greedy routing is preferable since the side paths are longer and affected by interference.

C. Description of the experiment

We consider two communication flows and use the nodes in the corners of the central area as endpoints. In our experiments, the node situated at P:29 sends packets to the node situated at A:14 and the node situated at P:14 sends packets to the node at A:27. We measure the throughput obtained when using geographic greedy routing and when splitting the traffic on two BGR paths.

We vary different parameters such as neighbor link quality threshold and the transmission rate. In order to create congestion, we synchronize the two senders. We use biases of +/-40° for the split paths; the values of the bias were chosen experimentally.

To filter out neighbors with poor links we use a threshold on the averaged link quality estimator (LQI). The maximum value of the 7 bit LQI filled by the communication driver for the CC2420 is 108, with 80's being the lower part of the working values. Because LQI refers to the reversed link that the packet will traverse, we would desire it as high as possible because it is known that, with high probability, strong links are bidirectional good.

D. Results and observations

Fig. 12 shows sample BGR paths on the testbed for three different biases. As link quality oscillates in time, for some paths (including greedy) we encountered two different versions. For illustrative purposes, we only present here one run (the LQI threshold is set to 84).

Tables 2-4 show the number of packets each destination receives out of 500 total packets sent, at three transmission rates: 40 packets/s (high contention), 33 packets/s (medium contention) and 20 packets/s (low contention). For each sending rate, we vary the neighbor selection threshold. Due to link quality oscillations, we sometimes measured different results for the same values of the parameters; these differences were caused by different path choices. We present both sets of results when the difference was large (as for LQI 86). Table 3 can be read as: when using only neighbors with averaged LQI value greater than 84, the total number packets received by the two destinations is 739 when sending on greedy paths and 869 when splitting the traffic on two BGR paths.

Overall, we observe that, at high transmission rates (40 packets per second), using multiple paths results in an increase of the delivery ratio from 11% up to 167%. The increase in throughput gets smaller as we decrease the transmission rate.

Table 5 shows the throughput variation as a function of retransmission count, when LQI is 84. As we increase the number of retransmissions, the throughput increases up to a point and then starts decreasing due to queue losses. Greedy is much more affected due to the higher packet frequency on the paths (actually in this case the two paths have a common hop). Also, greedy uses longer links that are worse on average and require more retransmissions (equivalent to increasing the transmission rate).

E. Short comments on the results

In our settings, greedy has best throughput at medium link quality thresholds. The most important reason is that links are still good and there are fewer nodes contending over the central broadcast domain. However, if the LQI threshold is low (80), the obtained throughput is highly variable, since some links are bad (i.e. almost all the packets are dropped). Because greedy forwarding chooses the most distant hop, the probability of encountering a bad link is high. For instance, at LQI 80, one communication is completely disrupted.

The throughput of the split BGR paths is less influenced by the LQI threshold. One reason is that distances traveled by a packet on each hop vary less for BGR hops as opposed to greedy. However, the maximum throughput is achieved at the same LQI as greedy. As we minimize the LQI threshold, links become worse and the interference affects the transmissions more. On the other hand, if we further increase the neighbor quality selection, the number of neighbor choices for BGR decreases (at LQI 86 some nodes have only one or two neighbors) and there is less control on the paths, which are longer and closer to each other. This is why the throughput decreases at the highest quality threshold.

Finally, there are runs for which sending on the two split paths results in a lower throughput (e.g. at 33 packets/s LQI 82). In such cases, the paths have a common point where interference is severe. This is not surprising, since the width of the testbed is quite narrow when compared to the radio range.

To conclude, our preliminary results suggest that, in practical scenarios, splitting the traffic in two BGR paths can provide better throughput performance than greedy routing in case of congestion and that the cost of interference is not high at lower transmission rates. However, a larger testbed is needed to evaluate the IPS and EPS congestion control mechanisms.

VI. RELATED WORK

To reflect this work's main contributions, we split the related work in two parts, describing congestion control in wireless networks and multipath algorithms, respectively.

A. Congestion Control for Wireless Sensor Networks

Congestion control in sensor networks for single flows has been initially explored by CODA [11] and FUSION [12]. Congestion is detected by sampling the wireless medium and checking that utilization is under a predefined threshold and by monitoring queue occupancy. In both [11] and [12], congestion alleviation is achieved with two mechanisms: open-loop, hop-by-hop mechanisms where nodes multiplicatively decrease sending rates towards congested neighbors and closed-loop source regulation, where a source sending at a high rate requires constant feedback from the destination to maintain its rate. Our mechanisms are multipath-based counterparts of the previously mentioned control mechanisms.

TABLE 2 – NUMBER OF PACKETS RECEIVED AT 40 PACKETS/S

Method	Dest	LQI 80	LQI 82	LQI 84	LQI 86
1 path	A:14	1	244	197	182
	A:27	267	418	211	219
	Sum	**267**	**662**	**408**	**401**
2 paths 40^0 bias	A:14	305	372	356	303
	A:27	409	364	349	271
	Sum	**714**	**737**	**705**	**574**
% increase		**167**	**11**	**72**	**42**

TABLE 3 – NUMBER OF PACKETS RECEIVED AT 33 PACKETS/S

Method	Dest	LQI 80	LQI 82	LQI 84	LQI 86
1 path	A:14	0	476	344	310/401
	A:27	341	469	395	331/476
	Sum	**341**	**945**	**739**	**641/847**
2 paths 40^0 bias	A:14	374	419	435	198/425
	A:27	380	415	433	391/328
	Sum	**754**	**835**	**869**	**589/811**
% increase		**120**	**-12**	**17**	**-8/-8**

TABLE 4 – NUMBER OF PACKETS RECEIVED AT 20 PACKETS/S

Method	Dest	LQI 80	LQI 82	LQI 84	LQI 86
1 path	A:14	0	493	474	394/394
	A:27	458	470	467	381/444
	Sum	**458**	**964**	**941**	**775/838**
2 paths 40^0 bias	A:14	469	475	461	237/404
	A:27	415	457	459	449/454
	Sum	**884**	**932**	**920**	**686/958**
% increase		**93**	**-3**	**-2**	**-12/14**

TABLE 5–THROUGHPUT VARIATION/RETRY COUNT AT 33 PACKETS/S

Retries	Dest	0	2	4	6
1 path	A:14	194	312	344	262
	A:27	212	329	395	314
	Sum	**406**	**641**	**739**	**576**
2 paths 40^0 bias	A:14	201	345	435	424
	A:27	234	360	433	403
	Sum	**435**	**705**	**869**	**827**
% increase		**7**	**9**	**17**	**43**

Reducing congestion with multipath routing has been addressed by several other works [27,17,14]. The first important difference of our work is that the location awareness assumption allows us to more easily create paths further away from congestion and releases the network from the burden of sending path creation and keep-alive messages and maintaining any state. Our solution avoids sending control messages in already congested areas.

Pham and Perreau [27] propose splitting the traffic from the start into multiple paths to achieve load balance and increase throughput. Unlike their solution we split reactively when congestion is detected, to avoid the additional costs of multipath routing when the network is not congested.

Authors of [17] and [14] also propose avoiding congestion through path diversity. In this paper, we propose a more general mechanism which works for both short-term, transient congestion and but also for long-term congestion through rate adaptation and dynamic path selection.

Finally, our method of signaling flow congestion and providing feedback to the source is similar to ECN [13]: a congestion bit in the packet header is set by nodes to signal congestion; to minimize energy consumption, these signals are filtered at the destination and sent to the source periodically.

B. *Geographic Multipath Routing*

The simplest instance of a multipath routing protocol in the context of geographic routing is waypoint routing: a packet is routed greedily towards a point or a list of points (landmarks) selected by the source, with the destination being the final point on the polygonal-line trajectory. As we have showed in section IV, the curved paths obtained by BGR are finer grained and have less overlap than single waypoint paths. Moreover, the waypoint location is usually represented on more than 1 byte (typically location has 2-4 bytes), and thus, BGR has lower overhead. This is important in wireless networks where energy, the scarcest resource, is used for sending data.

Trajectory Based Forwarding (TBF) [5] is the algorithm conceptually closest to BGR. TBF allows routing in networks with coordinate information on source specified trajectories. TBF resembles landmark routing, the difference being the way the landmarks are specified. Instead of embedding them in the packet, TBF embeds the equation of the trajectory into the packet and computes the landmarks at each hop. This approach is computationally expensive and quite complex, since at each step a part of the trajectory has to be simulated. TBF is generic and can achieve any trajectory, including approximations of the trajectories taken by the BGR packets. While TBF nodes solve an equation to determine the next hop, BGR can be viewed as a simple extension of greedy routing. Trajectories in BGR are not explicitly specified and their shape depends on the characteristics of the network (such as node density), in contrast to TBF's precise trajectories. In particular, this characteristic comes with the advantage of small communication and computation overhead (BGR uses one byte while only the data for the trajectory is typically over 6 bytes) and does not cause losses in efficiency.

Finally, there is a body of related work on achieving multiple paths without geographic knowledge [7-10,14]. In contrast to the solutions based on geographic routing, these proposals do not allow the end host to control the path selection process and have additional setup and state maintenance costs.

VII. Summary

In this paper, we have presented a solution that increases fairness and throughput in dense wireless networks. Our solution achieves its goals by using multipath geographic routing to find available resources in the network.

Biased Geographical Routing is our proposed solution for geographic multipath routing. The algorithm is simple and has low communication overhead; simulation results show that it compares favorably to other solutions and experimental deployment shows that it is usable in real life.

Running on top of BGR, we have proposed two algorithms, IPS (in-network packet scatter) and EPS (end-to-end packet scatter), that split a flow into multiple paths when it is experiencing congestion. IPS tries to solve transient congestion; when congestion persists EPS is activated. EPS performs rate control to minimize losses while maintaining high throughput. It uses a less aggressive congestion response for the non-greedy paths to gracefully capture resources available in the network.

Ns2 simulation results show that the combination IPS + EPS successfully improves fairness and throughput with small additional overhead. Experimental results confirm that multipath routing can indeed increase throughput; however, a larger testbed is needed to test EPS and IPS in practice.

References

[1] Pister K. S. J. Kahn J. M. and Boser B. E., "Smart Dust: Wireless Networks of Millimeter-Scale Sensor Nodes," in *Electronics Research Laboratory Research Summary*, 1999.

[2] Rao A. Ratnasamy S., Papadimitriou C., Shenker S., Stoica I., "Geographic Routing without Location Information," in Proc. of Mobicom, 2003.

[3] Moore D. Leonard J. Rus D. and Teller S., "Robust Distributed Network Localization with Noisy Range Measurements", in Proc. of Sensys 2004

[4] Fonseca R. Ratnasamy S., Zhao J., Ee C.T., Culler D., Shenker S., Stoica I., "Beacon Vector Routing: Scalable Point-to-Point Routing in Wireless Sensornets," in Proc. of NSDI, 2005.

[5] Niculescu D. Nath B., "Trajectory Based Forwarding and Its Applications," in Proc. of Mobicom, 2003.

[6] NS2 simulator, http://www.isi.edu/nsnam/ns/.

[7] Deb B. Bhatnagar S. and Nath B., "ReInForM: Reliable Information Forwarding using Multiple Paths in Sensor Networks," in Proc. of ICDCS, 2003.

[8] Ganesan D. Govindan R., Shenker S., and Estrin D., "Highly Resilient, Energy Efficient Multipath Routing in Wireless Sensor Networks," in ACM Mobile Computing and Communications Review, 2001.

[9] David D. Clark, Craig Partridge, Robert T. Braden, Bruce Davie, Sally Floyd, Van Jacobson, Dina Katabi, Greg Minshall, K.K. Ramakrishnan, Timothy Roscoe, Ion Stoica, John Wroclawski and Lixia Zhang, "Making the World (of Communications) a Different Place", in ACM Computer Communication Review, 35/3 , July 2005

[10] Nath S. Gibbons P.B., Seshan S., Anderson Z., "Synopsis Diffusion for Robust Aggregation in Sensor Networks," in Proc. Of SenSys, 2004.

[11] Wan C.Y. Eisenman S.B., Campbell A.T., "CODA: Congestion Detection and Avoidance in Sensor Networks," in Proc. of SenSys, 2003.

[24] [12] Hull B. Jamieson K., Balakrishnan H., "Mitigating Congestion in Wireless Sensor Networks," in Proc. of SenSys, 2004.

[13] Ramakrishnan K.K. Jain R., "A Binary Feedback Scheme for Congestion Avoidance in Computer Networks," in Transactions on Computer Systems, vol. 8, 1990.

[14] Kang J. Nath B., Zhang. Y., Yu S., "Adaptive Resource Control Scheme to Alleviate Congestion in Sensor Networks," in Proc. of Broadnets, 2004.

[15] Karp B. Kung H. T., "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," in Proc. of Mobicom, 2001.

[16] Kim Y.J. Govindan R., Karp B. and Shenker S., "Geographic Routing Made Practical," in Proc. of NSDI, 2005.

[17] D. A. Tran, H. Raghavendra, "Routing with Congestion Awareness and Adaptivity in Mobile Ad hoc Networks", in Proc. of WCNC 2005.

[18] Sinha P. Nandagopal T., Venkitaraman N., Sivakumar R., Bhargavan V., "A Reliable Transport Protocol for Wireless Wide-Area Networks.", in Proc. of Mobihoc, 2003

[19] TinyOs, http://www.tinyos.net.

[20] Levis P. Lee N., Welsh M., and Culler D., "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications," in Proc. of SenSys, 2003.

[21] The Berkeley Intel Research Mirage testbed, https://mirage.berkeley.intel-research.net/.

[22] MicaZ mote, http://www.xbow.com/Products/Product_pdf_files/ Wireless_pdf/MICAz_Datasheet.pdf.

[23] "The CC2420 radio data sheet, http://www.chipcon.com/files/CC2420_Data_Sheet_1_2.pdf

[25] Victor Bahl - Opportunities and Challenges of Community Wireless Networks – Keynote Talk, ETH's annual Mobile Information and Communications Systems' Zurich, 2004

[26] Ramakrishna Gummadi, Ramesh Govindan, Nupur Kothari, Brad Karp, Young-Jin Kim, Scott Shenker, "Reduced State Routing in the Internet", in Proc. Of Hotnets 2004

[27] Jinlyang Li, John Lannotti, Douglas S.J. De Couto, David R. Karger, Robert Morris – A Scalable Location Service for Geographic Ad Hoc Routing – in Proc. Of Mobicom, 2000

[28] Peter P. Pham and Sylvie Perreau, "Performance Analysis of Reactive Shortest Path and Multipath Routing Mechanism With Load Balance", in Proc. Of Infocom, 2003

[29] Piyush Gupta, P.R. Khumar, "Capacity of Wireless Networks", in IEEE Transactions on Information Theory, 46/2, March 2000