SPEED: A Real-Time Routing Protocol for Sensor Networks[1]

Tian He John A Stankovic Chenyang Lu Tarek Abdelzaher
Department of Computer Science
University of Virginia

Abstract

In this paper, we present a real-time communication protocol, called SPEED, for sensor networks. The protocol provides three types of real-time communication services, namely, real-time unicast, real-time area-multicast and real-time area-anycast. SPEED is specifically tailored to be a stateless, localized algorithm with minimal control overhead. End-to-end real-time communication guarantees are achieved using a novel combination of feedback control and non-deterministic QoS-aware geographic forwarding with a bounded hop count. SPEED is a highly efficient and scalable protocol for the sensor networks where node density is high while the resources of each node are scarce. Theoretical analysis and simulation experiments are provided to validate our claims.

1. Introduction

Many exciting results have been recently developed for large-scale ad hoc sensor networks. These networks can form the basis for many types of smart environments such as smart hospitals, battlefields, earthquake response systems, and learning environments.

The main function of sensor networks is data delivery. We distinguish three types of communication patterns associated with delivery of data in such networks. First, it is often the case that one part of the network detects some activity that it needs to report to a remote base station. This common mode of communication is called regular unicast. Alternatively, a base station may issue a command or query for an area in the ad hoc sensor network. For example, it may ask all the sensors in the region of a damaged nuclear plant to report radiation readings, or command all lights in a given area to turn on. This type of communication motivates a different routing service where one end-point of the route may be an area rather than an individual node. We call it area-multicast. Finally, since sensors often measure highly redundant information, in some situations it may be sufficient to have any node in an area respond. We call a routing service that provides such capability, area-anycast. SPEED provides the aforementioned three types of communication services.

In addition, since sensor networks are dealing with real world processes, it is often necessary that communication meet real-time constraints. To date, few results exist for ad hoc sensor networks that adequately address real-time requirements. In this paper we develop a protocol that provides real-time guarantees based on feedback control and stateless algorithms in large-scale networks. We evaluate SPEED via simulation using GloMoSim [17] and compare it to DSR[5], AODV[11] and GF[14]. The performance results show that SPEED reduces the number of packets that miss their end-to-end deadlines, reacts to transient congestion in the most stable manner, and has the least control packet overhead.

2. State of thea Art

A number of routing protocols (e.g., [2] [5] [8] [11] [12]) have been developed for *ad hoc* wireless networks. Sensor networks can be regarded as a sub-category of such networks, but with a number of different requirements.

First, in sensor networks, location is more important than a specific node ID. For example, tracking applications only care where the target is located, not the ID of the reporting node. In sensor networks, such position-awareness is necessary to make the sensor data meaningful. Therefore, it is natural to utilize location-aware routing. A set of location based routing algorithms have been proposed. For example, MFR [14] by Takagi et al. forwards a packet to the node that makes the most progress towards the destination. Finn [3] proposed a greedy geographic forwarding protocol with limited flooding to circumvent the voids inside the network. GPSR [7] by Karp and Kung used perimeter forwarding to get around voids. Another location-based routing algorithm is geographic distance routing (GEDIR) [14] which

[1] [1] This work was supported, in part by NSF grants CCR-0098269, the MURI award N00014-01-1-0576 from ONR, and the DAPRPA ITO office under the NEST project (grant number F336615-01-C-1905).

guarantees loop-free delivery in a collision-free network. Basagni, et. al. proposed a distance routing algorithm for mobility (DREAM)[2], in which each node periodically updates its location information by transmitting it to the all other nodes. The updating rate is set according to a distance effect in order to reduce the number of control packets. LAR [8] by Young-Bae Ko improves efficiency of the on-demand routing algorithm by restricting routing packet flooding in a certain "request zone."

SPEED also utilizes geographic location to make localized routing decisions. The difference is that SPEED is designed to handle congestion and provide delay guarantees, which were not the main goals of previous location-based routing protocols.

Reactive routing algorithms such as AODV[11], DSR[5] and TORA [10] maintain routing information for a small subset of possible destinations, namely those currently in use. If no route is available for a new destination, a route discovery process is invoked. Route discovery can lead to significant delays in a sensor network with a large network diameter (measured in multiples of radio radius). This limitation makes those algorithms less suitable for real-time applications.

Several research efforts have addressed providing quality of service guarantees in traditional wireless mobile networks. Lin [9] proposed an on-demand QoS routing scheme for multi-hop mobile networks. The scheme is based on a hop-by-hop resource reservation. This scheme works with small-scale mobile networks, where each node has enough memory to record the flow information and has high bandwidth to accommodate control overhead. In sensor networks, such schemes break due to scarce bandwidth and constrained memory. To the best of our knowledge, no algorithm has been specifically designed to provide real-time guarantees for sensor networks.

3. Design Goals

The goal of the SPEED algorithm is to provide three types of real-time communication services, namely, real-time unicast, real-time area-multicast and real-time area-anycast, for ad hoc sensor networks. In doing so, we satisfy the following additional design objectives.

1. **Stateless architecture.** The physical limitations of ad hoc sensor networks, such as large scale, high failure rate, and constrained memory capacity necessitate a stateless approach in which routers do not maintain much information about network topology and flow state. Routing-table based protocols, such as DSDV [12], are suitable for wireless networks with a relatively small number of nodes and large memories. It is hard to imagine, however, that each sensor node in a sensor network would be able to have thousands of routing entries that would be needed in state-based approaches. In contrast, SPEED maintains neither a routing table nor per-flow state. Thus, its memory requirements are minimal.
2. **Real-time guarantees.** Sensor networks are commonly used to monitor and control the physical world. To provide a meaningful service such as disaster and emergency surveillance, meeting real-time constraints is one of the basic requirements of such protocols. Algorithms using on-demand routing [5] [10] [11] are not designed to provide delay guarantees and may therefore fail to be suitable candidates for real-time applications. SPEED provides per-hop delay guarantees through a novel distributed feedback control scheme. Combining this feature with a simple scheme that bounds the number of hops from source to destination, SPEED achieves an end-to-end delay guarantee with small overhead.
3. **QoS routing and congestion management.** Most reactive routing protocols can find routes that avoid network hot spots during the route acquisition phase. Such protocols work very well when traffic patterns don't fluctuate very quickly during a session. They are less successful when congestion patterns change rapidly compared to session lifetime. When a route becomes congested, such protocols either suffer a delay or initiate another round of route discovery. SPEED uses a novel backpressure re-routing scheme to re-route packets around large-delay links with minimum control overhead.
4. **Traffic load balancing.** In sensor networks, the bandwidth is an extremely scarce resource compared to a wired network. Because of this, it is valuable to utilize several simultaneous paths to carry packets from the source to the destination. Most current solutions (e.g., [5][10][11]) don't utilize multiple paths, which leads to high queuing delays and unbalanced power consumption. Instead, SPEED uses non-deterministic forwarding to balance each single flow among multiple concurrent routes.
5. **Localized behavior.** Pure localized algorithms are those in which any action invoked by a node should not affect the whole system. In this sense, for algorithms such as AODV, DSR, and TORA this is not the case. In these protocols a node uses flooding to discover the new path. In sensor networks where thousands of nodes communicate with each other, broadcast storms may result in significant power consumption and possibly a network meltdown. Instead, all distributed operations in SPEED are localized to achieve high scalability.
6. **Loop-free routing.** Since SPEED is based on greedy geographic forwarding, it is inherently loop-free [15].

While SPEED does not use routing tables, SPEED does utilize location information to carry out routing.

This means that we assume that each node is location-aware. We also assume that the network is dense enough to allow greedy routing. In other words, at every hop of the path it is possible to find a next hop that is *closer* to the destination. In Appendix A we present a lower-bound analysis of node density that permits such greedy routing.

4. SPEED protocol

The SPEED protocol consists of the following components:

- The API
- Neighbor beacon exchange.
- Receive delay estimation
- The stateless geographic non-deterministic forwarding algorithm (SNGF).
- Neighborhood Feedback Loop.
- Backpressure Rerouting.
- Last mile local flooding.

These components are described in the subsequent sections, respectively.

4.1. Application API and Packet Format

The SPEED protocol provides four application-level API calls:

- AreaMulticastSend (position, radius, deadline, packet): This service identifies a destination area by its center position and radius. It guarantees that every node inside that area will receive a copy of the sent packet within the specified end-to-end deadline.
- AreaAnyCastSend (position, radius, deadline, packet): This service guarantees that at least one node inside the destination area receives the packet before the deadline.
- UincastSend(Global_ID,deadline,packet): In this service the node identified by Global_ID will receive the packet before the deadline.
- SpeedReceive(): this primitive permits nodes to accept packets targeted to them.

There is a single data packet format for the SPEED protocol. It contains the following major fields:

- PacketType: this field denotes the type of communication: AreaMulticast , AreaAnyCast or unicast.
- Global_ID: this field is only used in the unicast case to identify destination node.
- Destination area: Describes a 3D space with a center point and radius where the packets are targeted.
- TTL: Time To Live field is the hop limit used for last mile processing.
- Payload.

4.2. Neighbor Beacon Exchange

SPEED relies on neighborhood information to compute per-hop delays. Periodically and asynchronously, every node broadcasts a beacon packet to its neighbors. Each beacon packet has three fields:

(ID, Position, node *receive delay*)

where node receive delay is an average delay computed as described in the next subsection.

After receiving the beacon, a node saves the information in a neighborhood table, which will be used by other parts of the protocol.

In scarce bandwidth environments, the beaconing rate can't be too high; otherwise, it will impose significant overhead. To trade off control overhead for response time, in addition to periodic beacon transmissions, SPEED uses on-demand beacons to quickly identify traffic changes inside the network. With the help of on-demand beaconing, periodic beacon rates can be reduced to 1 per minute without degrading performance. Piggybacking [7] methods can also be exploited to reduce the beacon overhead. In our solution, beacon exchange doesn't introduce much overhead, but it does provide multiple benefits:

First, a beacon provides a way to detect node failure. If a neighbor entry is not refreshed after a certain time, it will be removed from the neighbor table. In this case packets will not be sent to that neighbor. If the un-responding neighbor later becomes available, the node will detect this and it will be added to the table.

Second, through beaconing, each node obtains the location information of its neighbors and this is used by SNGF to do geographic forwarding.

Third, the node *receive delay*, explained in the next section, is a metric that denotes the load of a neighbor. It is exchanged with beacons and used by the neighborhood feedback loop component of the protocol to provide real-time, single hop delay guarantees.

4.3. Receive Delay Estimation

We decided to use delay to approximate the load of a node. Nodes will transmit their delay to neighbors via a beacon packet. To do this each node keeps a neighbor table. Each entry inside the neighbor table has the following fields: (NeighborID, Position, ReceiveFromDelay, SendToDelay, ExpireTime). Since wireless channels are asymmetric, we separate the delay into two directions. The SendToDelay is the delay value received from the beacon message coming from neighbors and used to make routing decision by SNGF as discussed later. ReceiveFromDelay is estimated by measuring the real delay experienced by the data packet

in the MAC layer of the sender plus a propagation delay. This measurement can be done at the sender side without clock synchronization. Periodically the ReceiveFromDelays for each neighbor are averaged to compute a single node *receive delay*. If the new node-receive delay is larger or smaller than the previous node receive delay by a certain threshold, then an on demand beacon will be issued, otherwise there is no beacon.

The benefit of choosing an averaged node *receive delay* as the metric to denote load of a node is to reduce the channel capture effect [1] at the routing layer. The neighbor node that captures the channel will be notified of a longer delay through the beacon data than it actually experiences. As we can see in section 5.6, then it will re-route a fraction of packets to other forwarding nodes to reduce the load in this node. By using a single delay feedback, a node can indicate to all its neighbors to react cooperatively, such that the congestion can be reduced more fairly without starving any neighbor nodes.

4.4. Stateless Non-deterministic Geographic Forwarding (SNGF)

Before elaborating on SGNF, we introduce two definitions:

- The Neighbor Set of node *i*: This is the set of nodes that are inside the radio range R of node *i* and also at least *K* distance away from node *i*. Formally, $NS_i(K) = \{node \mid K \leq distance(node, node\ i) \leq R\}$.
- The Forwarding Candidate Set of node *i*: A set of nodes that belong to $NS_i(K)$ and are at least *K* distance closer to the destination. Formally, $FS_i(Destination) = \{node \in NS_i(K) \mid L - L\_next \geq K\}$ where *L* is the distance from node *i* to the destination and *L_next* is the distance from the next hop forwarding candidate to the destination. These nodes are inside the cross-hatched shaded area as shown in Figure 1. We can easily obtain $FS_i(Destination)$ by scanning the *NS* set of nodes once.

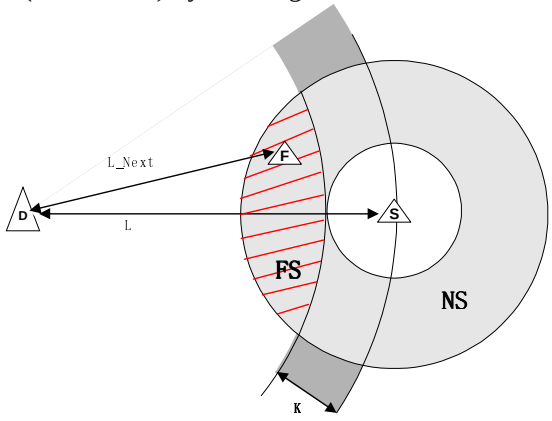

Figure 1. NS and FS definitions

It is worth noticing that the membership of the neighbor set only depends on the radio range *R* and distance *K*, but the membership of the forwarding set also depends on the position of destination nodes.

Since SPEED nodes only keep the Neighbor Set (NS), not the routing table, the memory requirement is only proportional to the number of neighbors, which can be ignored compared to the vast size of a sensor network.

Based on the destination of the packet and the current FS, the Stateless Non-deterministic geographic forwarding (SNGF) part of our protocol routes the packet according to following rules:

1. Packets are forwarded only to the nodes that belong to the $FS_i(Destination)$. If there is no node inside the $FS_i(Destination)$, packets are dropped. To reduce the chance of such a drop, we deduce a lower bound on node density that can virtually eliminate such drops from happening (appendix A).
2. SPEED divides the neighbor nodes inside $FS_i(Destination)$ into two groups. One group is the nodes that have SendToDelay less than a certain single hop delay D. The others are the nodes that have SendToDelay longer than a certain single hop delay D.
3. The forwarding candidate is chosen from the first group, and the neighbor node with highest relay SPEED $(|L - L\_next| / SendToDelay)$ has a higher probability to be chosen as the forwarding node. In our approach, we use a discrete exponential distribution to trade off between load balancing and optimal path length.
4. If there are no nodes belonging to the first group, a relay ratio is calculated based on the neighborhood Feedback Closed Loop, which we will discuss in more detail later. Whether a packet drop will really happen depends on whether a randomly generated number between (0,1) is bigger than the relay ratio. In SPEED a packet is dropped only when no downstream node can guarantee the single hop delay and an action must be taken to reduce the congestion. SPEED also provides back-pressure to the upper stream node which leads to automatic rerouting as we show in section 5.6.

SNGF provides two nice properties to meet our design goals. First, since it only sends packets to a node that is *K* nearer to the destination, it gives an upper bound on the number of hops the packet will travel to the destination $(L/K+1)$. This, together with a single hop delay guarantee provided by the Neighborhood Closed Loop Feedback routine and back-pressure rerouting (discussed later), can provide a steady state

end-to-end delay real-time guarantee for ad hoc sensor networks. Second, SNGF can balance the traffic and reduce congestion by dispersing the packet into a large relay area. (An analysis on load balance performance is provided in appendix B). This load balancing is valuable in a sensor network where the density of nodes is high and the communication bandwidth is scarce and shared. Also load balancing can balance the power consumption inside the sensor network to prevent some nodes from dying faster than others.

4.5. Neighborhood Feedback Loop (NFL)

The Neighborhood Closed Feedback Loop is the key component that guarantees the single hop delay. The Neighborhood Closed loop is an effective approach to maintaining system performance at a desired value. This has been shown in [13], where a low, targeted miss ratio of real-time tasks and a high utilization of the computational nodes are simultaneously achieved. Here we want to maintain a single hop delay below a certain value D, a performance goal the system builder desires.

Since SNGF give us an upper bound on the number of hops from the source to the destination, once a single hop delay D guarantee is provided, formally we can provide the following end-to-end real-time guarantee in SPEED:

- In steady state, $P\{$ *A packet misses its E2E deadline*$\} \leq \varepsilon$ (ε is the set point of the miss ratio, theoretically zero)

  where the E2E deadline is given as
  $D \times (Distance/K + 1)$.

We deem it a miss when a packet arrives at a node with a single hop delay longer than D, or if there is a loss due to collision, or a forced-drop. The percentage of such misses for the entire node is called the miss ratio. The responsibility of the neighborhood closed feedback loop is to force such a miss ratio to converge to the set point, namely zero.

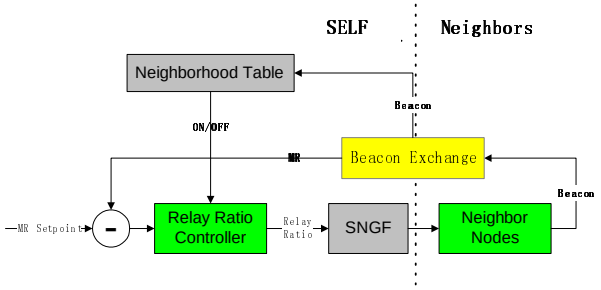

Figure 2. Neighborhood Feedback Loop (NFL)

As shown in Figure 2, the feedback loop is naturally established through the neighbor beacon exchange we mention in section 5.2. The Relay Ratio controller calculates the relay ratio and feeds that back into SNGF where a drop or relay action is made. The Relay ratio controller currently implemented is a simple multiple inputs single output (MISO) proportional controller that takes the miss ratios of all next-hop neighbors as inputs and proportionally calculates the relay ratio as output to the SNGF.

The Relay Ratio controller will be activated only when all nodes inside the forwarding set have a SendToDelay delay larger than the desired single hop delay D, which means that the neighborhood closed feedback loop will not be invoked unless there is no way to forward the packet to a non-busy node and a drop is absolutely necessary to guarantee the single hop delay. As we will see later in the backpressure rerouting, such a scheme enforces that re-routing has a higher priority than dropping. In other words, SPEED will not drop a packet as long as there is another path that can meet the delay requirements.

4.6. Back-Pressure Rerouting

Back-pressure re-routing is naturally generated from collaboration of neighbor feedback loop (NFL) routines as well as the stateless non-deterministic geographic forwarding (SNGF) part of SPEED without any additional control packets required. To be more explicit, we introduce this scheme with an example (Figure 3).

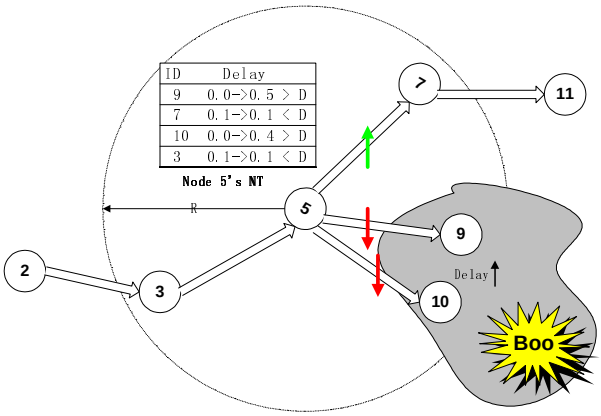

Figure 3. Back-pressure rerouting case one

Suppose in the lower-right area, some heavy traffic appears, which leads to high delay in nodes 9 and 10. Through beacon exchange node 5 will detect that nodes 9 and 10 have a higher delay. Since SNGF will reduce the probability of selecting nodes 9 and 10 as forwarding candidates, it will reduce the congestion around nodes 9 and 10. Since all neighbors of 9 and 10 will react the same as node 5, eventually nodes 9 and 10 will adjust their delay below the single hop deadline through Back-pressure re-routing.

Things are not always as simple as in previous case. A more severe case could happen when all the forwarding neighbors of node 5 are also congested as shown in figure 4.

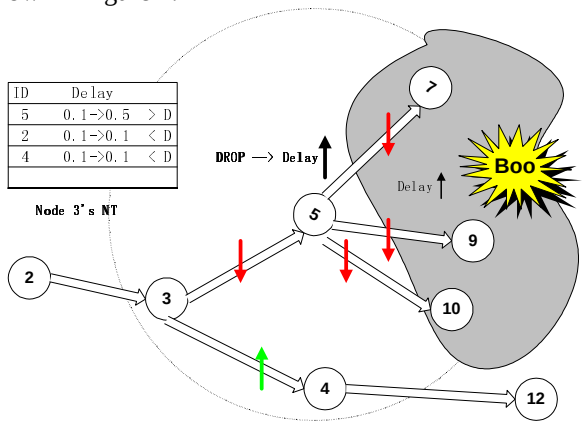

Figure 4. Back-pressure rerouting case 2

In this case, the neighborhood feedback loop is activated to assist backpressure re-routing. In node 5 a certain percent of packets will be dropped and when dropped they count as a packet with delay D in terms of computing the delay at this node. The increased average delay will be detected by upstream nodes, here node 3, which will react in the same way as above. If, unfortunately, node 3 is in the same situation as node 5, further backpressure will be imposed on node 2. In the extreme case, the whole network is congested and the backpressure will proceed upstream until it reaches the source, where the source will quench the traffic.

Backpressure rerouting is the first choice used by SPEED to reduce the congestion inside the network. In this case no packet need be sacrificed. A drop through the feedback-closed loop is only necessary when the situation is poor and there is no alternative except to drop the packet.

4.7. Last Mile Process

Since SPEED is targeted at sensor networks where the ID of nodes is not necessary, we don't care about the ID's of individual sensor nodes; instead we only care about the location of where the sensor data is generated.

The Last mile process is so called because only when the packet enters into the destination area will such a function be activated. All previous packet relays are controlled by the SNGF routine as mentioned before. The nodes inside the destination area comply with the last mile rules and will not be constrained by the K distance limitation used in SNGF.

The Last mile process provides two novel services that fit the scenario of sensor networks: Area-multicast and Area-anycast. The area here is defined by a center-point (x,y,z) and a radius, which mean it's a sphere. In general, more complex area definitions can be made without jeopardizing the design of the Last mile process.

The current implementation of the last mile process is relatively simple. Nodes can differentiate the packet type by the PacketType field mentioned in section 4.1. If it's an anycast packet, the nodes inside the destination area will deliver the packet to the transportation layer without relay. If it's a multicast type, the nodes inside the destination area which first receive the packet coming from the outside of the destination area will set a TTL. This allows the packet to survive the diameter of the destination area and be broadcast to the correct radius. Other nodes inside the destination area will keep a copy of the packet and re-broadcast it. The nodes that are outside the destination area will just ignore it. The last mile process for unicast is nearly the same as multicast, except only the node with that global_ID will deliver the packet to the transport layer. If the location directory service is precise, we can expect the additional flooding overhead for the unicast packets to be small.

5. Experimentation and Evaluation

5.1. Simulation Environment

We simulate SPEED on GloMoSim [17], which is a scalable discrete-event simulator developed by UCLA. This software provides high fidelity simulation for wireless communication with detailed propagation, radio and MAC layers simulation. Table 1 describes the detailed setup for the simulator. The communication parameters are chosen from UCB's TinyOS implementation [4] and its mote specification.

Transportation Layer	UDP
Routing	AODV, DSR, GF, SPEED
Network Layer	IP
MAC Layer	802.11
Radio Layer	RADIO-ACCNOISE
Propagation model	TWO-RAY
Bandwidth	200Kb/s
Payload size	32 Byte
TERRAIN	(2000m, 2000m)
Node number	100
Node placement	Uniform
Radio Range	377m

Table 1. Simulation settings

We discovered that using the CSMA/CA protocol available in GloMoSim introduces a high packet loss ratio due to the hidden terminal problems that are common in congested multi-hop sensor networks,

where collisions tend to be heavy. So instead we used 802.11 as an alterative MAC layer protocol. Since our protocol stack for the physical motes will not use UDP/IP, which has 28 bytes header overhead, to adjust for these large overheads, compared to the small payload size, another change we made was that the bandwidth was chosen to be double the current mote capability.

5.2. Congestion Avoidance

In a sensor network, where node density is high and bandwidth is scarce, traffic hot spots can be easily created. In turn, such hot spots may interfere with real-time guarantees of critical traffic in the network. To alleviate this problem, SPEED supports a combination of QoS-aware routing and back-pressure congestion control suitable for sensor networks.

To test the aforementioned capabilities, we used a cross traffic scenario, where 4 nodes at the upper left corner of the terrain send periodic data to a base station at the lower right corner and 4 nodes at the lower left corner send periodic data to the base station at the upper right corner. The average hop count between the node and base station is about 10 hops. Each node generates 1 CBR flow with a rate of 0.5 packet/second. To create congestion, at time 40 seconds, one heavy flow is created between two randomly chosen nodes in the middle of the terrain with a rate of 50 packets/second. This flow disappears at time 120 seconds into the run. This heavy flow introduces a step change into the system, which is an abrupt change that stress-tests SPEED adaptation capabilities to reveal its transient-state response.

We repeated the experiments with the same traffic load settings but different routing algorithms to compare SPEED to DSR, AODV, and GF. For each routing algorithm we report the average performance of 16 runs with different random seeds. Figures 5.A, 5.B 5.C and 5.D plot the end-to-end (E2E) delay profiles for the four different routing algorithms. At each point in time, we average the E2E delays of all the packets from the 128 flows (16 runs with 8 flows each). For legibility, any average E2E delay larger than 300 milliseconds is drawn as 300 milliseconds in the figures. Separately, Figure 6 uses statistical comparative boxplots to show the number of packets lost during the congestion. In these boxplots, the line in the middle of the box denotes the median value of the data set. The bottom and top of the box are the 25th and 75th percentiles. T-shaped lines here denote the maximum and minimum value of the data set.

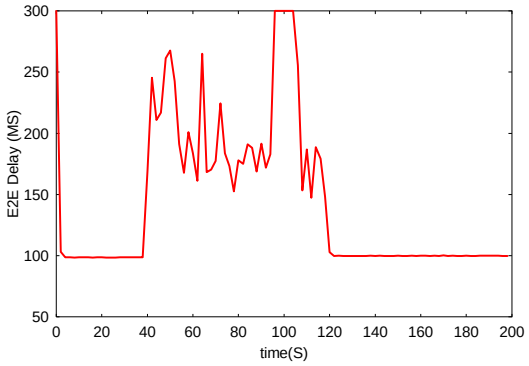

Figure 5.A. E2E delay profile of DSR

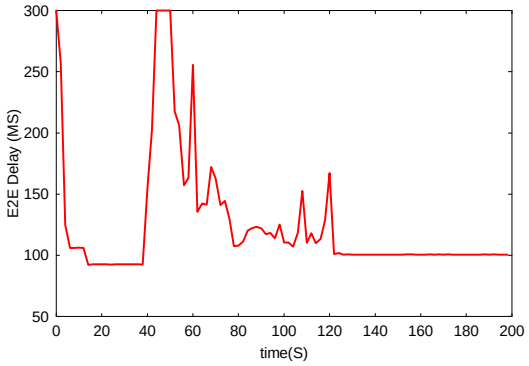

Figure 5.B. E2E delay profile of AODV

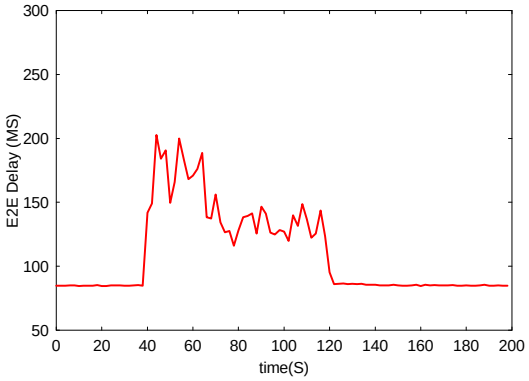

Figure 5.C. E2E delay profile of GF

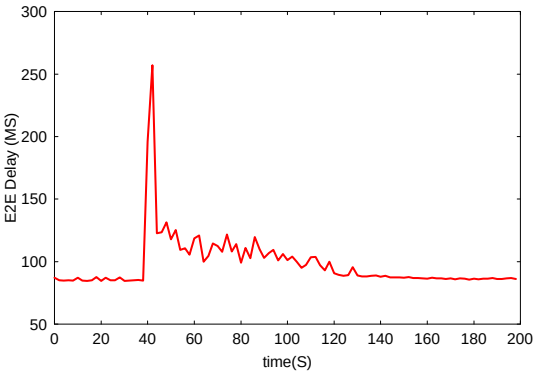

Figure 5.D E2E delay profile of SPEED

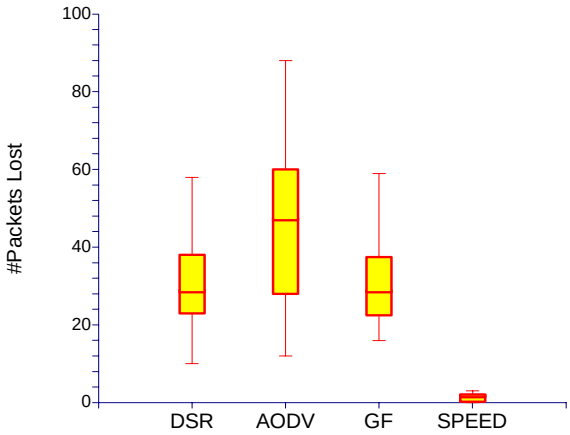

Figure 6. Number of packets lost during congestion

When DSR and AODV begin, they need to perform route acquisition to discover a path to the destination, causing a large delay (Figures 5.A and 5.B) for the first packet (10 ~ 20 times longer than the delay of the following packets). On the other hand GF and SPEED don't suffer from this feature. In these protocols, the first packet has same delay as the following packets. Without an initial delay cost, GF and SPEED are more suitable for real-time applications like acoustic tracking where the base station sends the actuation commands to the sensor group, which is dynamically changing as the target moves. In such a scenario, DSR and AODV need to perform route acquisition repeatedly in order to track the target.

In general, both GF and SPEED tend to forward the packet at each step as close to the destination as possible, which leads to less overall E2E delay than for DSR and AODV (See Figures 5.C and 5.D)

At time 40 sec when the heavy flow appears, the system enters into a transient state of congestion. Each routing algorithm responds differently. The key reasons for the performance profiles in the graphs of Figure 5 are 1) DSR, AODV and GF only respond to severe congestion that leads to link failures (i.e., when multiple retransmission fails at the MAC layer). They are insensitive to long delay as long as no link failures occur. 2) Their routing decisions are not based on the delays on the link, and therefore may congest a particular receiver even though it has long delays. 3) DSR and AODV flood the network when the network is already congested. This causes many packets to be dropped during the transient state. Below, we elaborate on details obtained from the simulation traces that describe the simulated behavior of each protocol under very heavy load. It should be noted that DSR, AODV, and GF (much like IP) were not designed with congestion control in mind. They delegate this function to a transport layer protocol. In contrast, SPEED integrates QoS routing with back pressure congestion control to modulate the traffic injected into the network and routes that traffic around the network hot spots.

DSR detects the congestion via the MAC layer notification of failure. It doesn't invoke another round of route acquisition immediately; instead it salvages the packet that the MAC failed to send and tries another known route. Since the policy DSR uses to choose a new route doesn't explicitly consider load information, this new route can also be congested, which means that it's difficult for DSR to find a good route. If congestion makes all attempted routes appear to have failed and if route acquisition is invoked, the network is flooded, which is not good for an already congested network.

AODV reacts to the congestion more aggressively. Upon receiving notification of a MAC failure, a node drops the packet that leads to the MAC layer failure and notifies all upstream nodes about the failure including the source node. The source initiates a new round of route acquisition. Hence, the AODV frequently repeats the route acquisition process (seen by the large number of control packets) when the network is severely congested. Most packets are dropped during that phase as shown in Figure 6.

GF responds to congestion by deleting the neighbor node that failed to receive the packet and by choosing another node that is the second nearest to the destination. The routing decision is based solely on distance and does not consider delay. Thus, it does not fully avoid the congestion problem as shown in the figure 5.C.

SPEED reacts differently in face of congestion. Through neighborhood feedback loops and back pressure rerouting, SPEED is sensitive to traffic changes. As shown in figure 5.D, at time 40, SPEED has longer E2E delays (overshoot) for several packets due to the backpressure rerouting. This is because at time 40 several packets suddenly enter the congested area and suffer a long delay; this long delay is detected by the receiving node and that information is fed back to the sender via the on-demand beacon. After that, SPEED finds un-congested routes to the destination. Since the new route has more hops to the destination, the E2E delay is longer than the initial path. At time 120, when congestion disappears, SPEED switches back to the initial paths with shorter delay.

5.3. E2E Deadline Miss Ratio

The deadline miss ratio is the most important metric in soft real-time systems. We set the E2E deadline equal 200 milliseconds, which is 10 times the delay of a single hop deadline. The results shown in Figure 7 are the summary of 16 runs with different seeds. AODV and DSR don't perform well in the face of congestion mainly because both algorithms flood the network in order to discover a new path when

congestion leads to a link failure, which makes the situation even worse. GF and SPEED treat congestion differently from AODV and DSR. GF deletes the failed neighbor from the table and chooses another forwarding node without considering the delay. This leads to about a 9% miss ratio (Figure 7). SPEED takes deadlines into account, and this leads to a very small miss-ratio (about 2%). A key observation here is that purely localized algorithms without flooding win when traffic congestion gets worse.

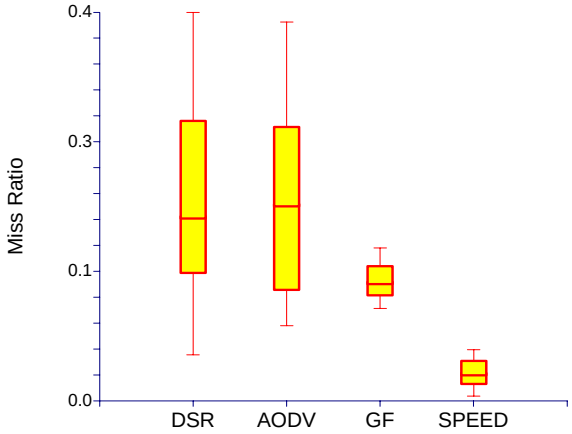

Figure 7. MissRatios during the congestion

5.4. Control Packet Comparison

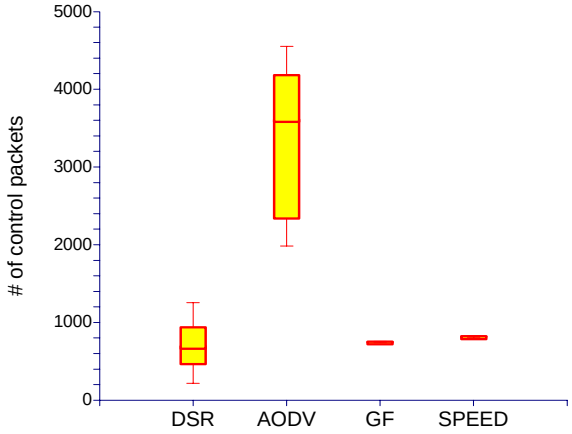

Figure 8. Control packet overhead comparison

All four routing algorithms studied use a relatively low number of control packets. Most control packets in DSR and AODV are used in route acquisition. Because AODV performs several rounds of route acquisition before settling down to a better route, it requires the most control packets. DSR uses a route cache extensively, so it can do route discovery and maintenance at a much lower cost than AODV. The only control packets used in GF and SPEED are beacon exchange packets. In addition to periodically sending beacons, SPEED uses on-demand beacons to notify neighbors of the congestion situation, which costs SPEED more control packets than GF (see Figure 8). Another observation is that the number of control packets in DSR and AODV vary a lot in different runs, while GF and SPEED use nearly constant number of control packet during each run. The key reason for this is the way DSR and AODV treat severe congestion as link failures and react by flooding the network.

6. Conclusion

Many excellent protocols have been developed for ad hoc networks. However, ad hoc sensor networks have additional requirements that were not specifically addressed. These include real-time requirements and nodes which are severely constrained in computing power, bandwidth, and memory. SPEED uses local feedback control to guarantee a per node delay in steady state and a combination of geographic forwarding and backpressure to bound the number of hops and react to changing loads and congestion. This improves the end-to-end delay and provides good response to congestion. Our simulations on GloMoSim demonstrate SPEED's improved performance compared to DSR, AODV and GF. Since these protocols were not originally developed for real-time sensor networks, it is not surprising that they don't work well in such situations. We were able to develop a new protocol that does meet the requirements of ad hoc sensor networks in real-time situations.

References

[1] Almes, G. T., Lazowska, E. D., i. The Behavior of Ethernet-Like Computer Communications Networks *ACM SIGCOMM* 1979

[2] S. Basagni and et. al. A Distance Routing Effect Algorithm for Mobility (DREAM). In *ACM/IEEE Int. Conf. on Mobile Computing and Networking (MobiCom '98*), October 1998

[3] Gregory G. Finn. Routing and Addressing Problems in Large Metropolitan-scale Internetworks. *Technical Report ISI/RR-87-180, USC/ISI*, March 1987.

[4] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, System Architecture Directions for Network Sensors. *ASPLOS 2000*.

[5] David B. Johnson and David A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Mobile Computing*, edited by Tomasz Imielinski and Hank Korth, Chapter 5, pages 153-181, Kluwer Academic Publishers, 1996.

[6] D. Kandlur, K. G. Shin, and D. Ferrari. Real-time

Communication in Multi-hop Networks, *IEEE Trans. on Parallel and Distributed Systems*, October 1994, pp. 1044-1056.

[7] Brad Karp and H. T. Kung. GPSR: Greedy PerimeterStateless Routing for Wireless Networks. In *Proc. AC Mobicom*, Boston, MA, 2000.

[8] Young-Bae Ko and Nitin H. Vaidya. Location-Aided Routing(LAR) in Mobile Ad Hoc Networks. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 1998)*, ACM, Dallas, TX, October 1998

[9] Lin, C.R. On-Demand QoS Routing in Multihop Mobile Networks. *INFOCOM 2001. Proceedings. IEEE* , Volume: 3 , 2001 Page(s): 1735 -1744 vol.3

[10] V.D. Park and M.S. Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks In *Proceedings of IEEE IN-FOCOM'97*, Kobe, Japan, Apr. 1997, pp. 1405-1413.

[11] C. E. Perkins and E. M. Royer, Ad-hoc On Demand Distance Vector Routing. In *2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99)*, New Orleans, Louisiana, February 1999.

[12] Charles E. Perkins and Pravin Bhagwat, Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for Mobile Computers, in *SIGCOMM Symposium on Communications Architectures and Protocols*, (London, UK), pp. 212-225, Sept. 1994.

[13] J. A. Stankovic, Tian He, T. F. Abdelzaher, M. Marley, G. Tao, S. Son, and C. Lu. Feedback Control Scheduling in Distributed Systems, *IEEE Real-Time Systems Symposium*, London, UK, December 2001.

[14] H.Takagi and L.Kleinrock. Optimal Transmission Ranges For Randomly Distributed Packet Radio Terminals. *IEEE Trans. on Communication*, 32(3):246-257, March 1984

[15] I. Stojmenovic and X. Lin. GEDIR: Loop-Free Location Based Routing in Wireless Networks, *IASTED Int. Conf. on Parallel and Distributed Computing and Systems*, Nov. 3-6, 1999, Boston, MA, USA.

[16] Ya Xu, John Heidemann, and Deborah Estrin. Geography-informed Energy Conservation for Ad Hoc Routing, In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2001)*, Rome, Italy, July 16-21, 2001

[17] Xiang Zeng, Rajive Bagrodia, and Mario Gerla , GloMoSim: a Library for Parallel Simulation of Large-scale Wireless Networks. In *Proceedings of the 12th Workshop on Parallel and Distributed Simulations -- PADS '98*, May 26-29, 1998 in Banff, Alberta

Appendix A: LowerBound of Node Density

One basic assumption of sensor networks is their high node density. It is an interesting research issue to determine the impact of node density on routing performance. Specifically, in the SPEED algorithm, we want to find the lower bound of node density that can guarantee that there is no void space that can prevent a greedy geographic forwarding step from happening.

In the SPEED algorithm, a node only sends packets to nodes that at least K distance nearer to the destination. The area where such qualified nodes reside is called the forwarding area. Assume the nodes are uniformly distributed inside the system, the larger the size of the forwarding area, the higher is the probability that there will be a candidate to be chosen. In SPEED such a forwarding area size is not constant; it is depends on how far away the sending node is from the destination node.

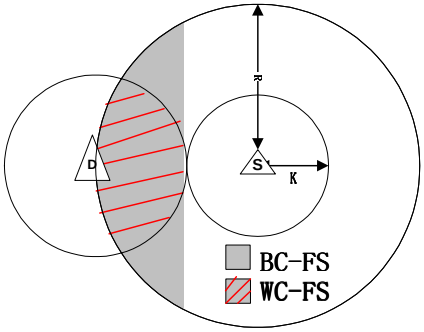

Figure 9. Forwarding Areas

When the destination node is infinitely far away from the sending node, the forwarding area will be the largest (Best Case Forwarding Size) and when the destination node is exactly R away from sending node (Figure 9), the available forwarding size is the worst case forwarding size (WCFS). For guaranteeing purposes, we only consider the worst case, even though most of the time the forwarding size is nearer to the best case. In the worst case, the forwarding size is calculated by following formula (1):

$$WCFS = (R-K)^2 \cos^{-1}\frac{R-K}{2R} + R^2 \cos^{-1}\left[1-\frac{(R-K)^2}{2R^2}\right] - \frac{1}{2}(R-K)\sqrt{4R^2-(R-K)^2}$$

which depends on the radio range R and the single step bound K. Now, we consider the worst case forwarding area. We desire to know the lower bound of node density that satisfies the following condition:

P (At least one node resides in the forwarding area for given parameter R and K) >= 1- ε (2)

Assuming a uniform distribution, according to (2) the following condition must hold: (the size of the area covered by the sensor network is denoted by AreaSize >> WCFS)

$$(1-\frac{WCFS}{AreaSize})^{AreaSize \times Density} \le \varepsilon \quad (3)$$

Since the left hand side of the equation is a monotonically increasing function when the AreaSize increases and monotonically decreasing when node density increases, the lower bound of the node density is achieved when AreaSize is infinite:

$$\lim(1-\frac{WCFS}{AreaSize})^{AreaSize \times Density} = e^{-WCFS \times Density} \le \varepsilon$$

Figure 10 shows the lower bound of node density that can probabilistically guarantee that there is no void inside a designated forwarding area (size is calculated by (1)). For example, from the figure 10, when K = 1/2R, if the node density is above 30 nodes per Radius-square, 99.9999% of time, the SPEED algorithm can find a forwarding candidate. Accordingly, if the diameter of the system is 100 hops, 99.990% of the time, the SPEED algorithm can find a 100-hop greedy geographic forwarding path to the destination.

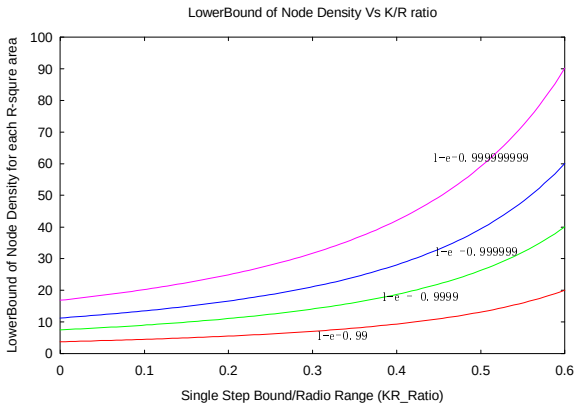

Figure 10. Lower bound node densities

Appendix B. Load Balancing in SPEED

SPEED uses non-deterministic geographic forwarding to spread out the traffic, in order to avoid bottlenecks and balance power consumption. In this appendix, we show some theoretical properties of load balancing in SPEED. Specifically, we derive the maximum *flow area* (Figure 11). This area is defined as the largest area, which may contain nodes that *forward* packets of the same SPEED flow. We show that the maximum flow area is a function of the distance L between the sender and receiver, the wireless transmission radius R, and the step K.

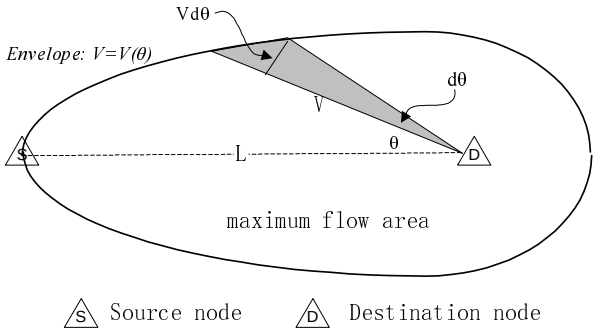

Figure 11. Maximum Flow Areas

To simplify the analysis, let us assume that the distance L between sender and receiver is much larger than the step K and radius R. Let us call the curve that envelopes the maximum flow area, $V(\theta)$ (expressed in polar coordinates). Without loss of generality, let our coordinates be centered at the destination, such that any point on the envelope is described by the pair $(\theta, V(\theta))$ specifying its angle and radial distance from the destination respectively, as shown in Figure 12.

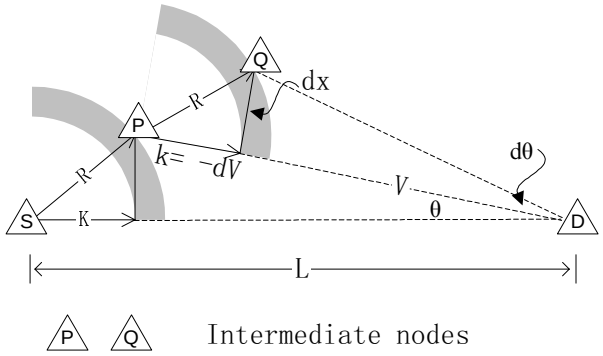

Figure 12. Forwarding Detail

Consider the point $(\theta, V(\theta))$ shown in figure 12. Let us call it envelope point P. To derive an expression for the envelope, we first show how to derive, given point P, the location of the next point on the envelope. For this purpose, informally, it is enough to ask the following question: assuming a node located at P forwards a packet of the flow, what is the furthest that the next hop of this packet can be from the line that joins the source and destination? The answer is obtained by intersecting the locus of all points that are one transmission radius away from P with the locus of all points that are K units closer to the destination. Note that, as long as $V >> K$, the latter locus can be approximated by a line perpendicular to V. The corresponding intersection point is labeled Q in Figure 11. Moving from P to Q, θ changes by $d\theta$ and V by dV

= -*K*. We can now derive a relation between $d\theta$ and *dV*. Since *dV* << *V*, observe from Figure 11 that:

$$d\theta = dx/V$$

where $dx = \sqrt{R^2 - K^2} = K\sqrt{(R/K)^2 - 1}$

Since $K = -dV$ we get:

$dx = -\sqrt{(R/K)^2 - 1}\, dV$, and:

$$d\theta = -\frac{\sqrt{(R/K)^2 - 1}}{V} dV$$

Integrating

$$\int_0^{\theta} d\theta = -\int_L^V \frac{\sqrt{(R/K)^2 - 1}}{V} dV$$

Thus, $\theta = \sqrt{(R/K)^2 - 1} \ln\frac{L}{V}$

Alternatively: $V = L\, e^{\frac{-\theta}{\sqrt{(R/K)^2 - 1}}}$

The above is the equation of the envelope. The area enclosed by the envelope is twice the area between one side of the envelope and the horizontal axis. In the polar coordinate framework this amounts to:

$$Area = \int dArea = 2\int_0^{\pi} \frac{1}{2} V^2 d\theta = L^2 \int_0^{\pi} e^{\frac{-2\theta}{\sqrt{(R/K)^2 - 1}}} d\theta$$

Hence:

$$Area = \frac{L^2}{2}\sqrt{(\frac{R}{K})^2 - 1}\left(1 - e^{\frac{-2\pi}{\sqrt{(R/K)^2 - 1}}}\right)$$

Obviously the above expression is approximate since *V* may eventually become small enough such that *V* >> *K* no longer holds, thus invalidating the assumption we used in the derivation. This part of the curve, however, by definition is very close to the destination. The area under it is only a small fraction of the total area computed. Hence, it need not be accurately determined for our overall area estimate to remain a good approximation.

In particular, it is easy to verify (by applying L'Opital's rule) that when $K \to 0$, then *Area* defined by the above equation is $Pi*L^2$ which is the area of a circle of radius *L*. This is intuitively correct, because when $K \to 0$, in the worst case a packet can circle around the destination without approaching the destination radially. The area of that circle is $Pi*L^2$. Note also that when $K \to R$, then $Area \to 0$. This again is intuitively correct, since the only way a packet can advance *R* towards the destination on every transmission is if that transmission is precisely along the straight line towards the destination.

The derivation of the flow area is important for analyzing network load and specifically for analyzing interference between different flows. Flows whose area boundaries are less than one interference radius apart will interfere with each other. Analysis of such interference can be used for admission control and congestions avoidance in SPEED. These issues will be addressed in more detail in later publications.