SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks

Tian He[a]   John A Stankovic[a]   Chenyang Lu[b]   Tarek Abdelzaher[a]

[a] *Department of Computer Science*
University of Virginia
{tianhe, stankovic, zaher}@cs.virginia.edu

[b] *Department of Computer Science & Engineering*
Washington University in St Louis
lu@cs.wustl.edu

Abstract

In this paper, we present a real-time communication protocol for sensor networks, called SPEED. 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 soft real-time communication is achieved by maintaining a desired delivery speed across the sensor network through a novel combination of feedback control and non-deterministic geographic forwarding. SPEED is a highly efficient and scalable protocol for sensor networks where the resources of each node are scarce. Theoretical analysis, simulation experiments and a real implementation on Berkeley motes are provided to validate our claims.

1. Introduction

Many exciting results have been recently developed for large-scale 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. While these potential applications remain diverse, one commonality they all share is the need for an efficient and robust routing protocol.

The main function of sensor networks is data delivery. We distinguish three types of communication patterns associated with the delivery of data in such networks. First, it is often the case that one part of a network detects some activity that needs to be reported to a remote base station. This type of communication is called unicast. Alternatively, a base station may issue a command or query to an area in the sensor networks. For example, it may ask all 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 this 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 afore-mentioned three types of communication services.

Since sensor networks deal with real world, it is often necessary for communication to meet real-time constraints. In surveillance systems, for example, communication delays within sensing and actuating loops directly affect the quality of tracking. To date, few results exist for sensor networks that adequately address real-time requirements. In this paper we develop a protocol SPEED that supports soft real-time communication based on feedback control and stateless algorithms for large-scale sensor networks. We evaluate SPEED via simulation using GloMoSim [15] and compare it to five other ad hoc routing protocols: DSR [5], AODV [10], GF [13] and two scaled down versions of SPEED. The performance results show that SPEED 1) reduces the number of packets that miss their end-to-end deadlines, 2) reacts to transient congestion in the most stable manner, and 3) efficiently handles voids [6] with minimal control overhead. We also implement SPEED on the Berkeley motes [4]. The results show that SPEED helps balance the traffic load to increase the system lifetime.

2. State of the Art

Several routing protocols 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.

In sensor networks, location is more important than a specific node's ID. For example, tracking applications only care where a target is located, not the ID of the reporting node. In sensor networks, such location-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. Finn [2] proposed a greedy geographic forwarding protocol with limited flooding to circumvent the voids inside the network. GPSR [6] by Karp and Kung use perimeter forwarding to get around voids. Geographic distance routing (GEDIR) [13] guarantees loop-free delivery in a collision-free network. LAR [7] by Young-Bae Ko improves the efficiency of the on-demand routing algorithms by restricting routing packet flooding in a specified "request zone."

SPEED also utilizes geographic location to make localized routing decisions. The difference is that SPEED is designed to handle congestion and provide a soft real-time communication service, which are not the main goals of previous location-based routing protocols. Moreover, SPEED provides an alternative solution to handle voids other than approaches based on planar graph traversal [6] and limited flooding [2].

Several real-time protocols have been proposed for sensor networks. SWAN [1] uses feedback information from the MAC layer to regulate the transmission rate of non-real-time TCP traffic in order to sustain real-time UDP traffic. RAP [9] uses velocity monotonic scheduling to prioritize real-time traffic and enforces such prioritization through a differentiated MAC Layer. Woo and Culler [14] proposed an adaptive MAC layer rate control to achieve fairness among nodes with different distances to the base station. All of these algorithms work well by locally degrading a certain portion of the traffic. However, this kind of local MAC layer adaptation cannot handle long-term congestion where routing assistance is necessary to divert traffic away from any hotspot. SPEED provides a combination of MAC layer and network layer adaptation that effectively deals with such issues. To the best of our knowledge, no routing algorithm has been specifically designed to provide soft real-time guarantees for sensor networks.

Reactive routing algorithms such as AODV [10] and DSR [5] 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 broadcasts can lead to significant delays in a sensor network with a large network diameter. This limitation makes on-demand algorithms less suitable for real-time applications.

3. Design Goals

Our design is inspired by the observation that unlike wired networks, where the delay is independent of the physical distance between the source and destination, in multi-hop wireless sensor networks, the end-to-end delay depends on not only single hop delay, but also on the distance a packet travels. In view of this, the key design goal of the SPEED algorithm is to support a soft real-time communication service with a desired delivery *speed* across the sensor network, so that end-to-end delay is proportional to the distance between the source and destination. It should be noted that delivery speed refers to the approaching rate along a straight line from the source toward the destination. Unless the packet is routed exactly along that straight line, delivery speed is smaller than the actual speed of the packet in the network. For example, if the packet is routed in the opposite direction from the destination, its speed is negative. Our algorithm ensures that this condition never occurs.

Upon this soft real-time delivery service, SPEED provides three types of real-time communication services, namely, real-time unicast, real-time area-multicast and real-time area-anycast, for sensor networks. In doing so, SPEED satisfies the following design objectives.

1. **Stateless Architecture**. The physical limitations of sensor networks, such as large scale, high failure rate, and constrained memory capacity necessitate a stateless approach. SPEED only maintains immediate neighbor information. It doesn't require a routing table as in DSDV [11] nor per-destination states as in AODV [10]. Thus, its memory requirements are minimal.
2. **Soft Real-Time.** Sensor networks are commonly used to monitor and control the physical world. SPEED provides a uniform delivery speed across the sensor network to meet the requirement of real-time applications such as disaster and emergency surveillance in sensor networks.
3. **Minimum MAC Layer Support.** SPEED doesn't require real-time or QoS aware MAC support. The feedback control scheme employed in SPEED allows it to be compatible with all existing best effort MAC layers.
4. **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 well when traffic patterns don't fluctuate during a session. However, these protocols (e.g. [5]) are less successful when congestion patterns change rapidly compared to the session lifetime. When a route becomes congested, such protocols either suffer a delay or initiate another round of route discovery. As a solution, SPEED uses a novel backpressure re-routing scheme to re-route packets around large-delay links with minimum control overhead.
5. **Traffic Load Balancing.** In sensor networks, the bandwidth and energy are scarce resources 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. SPEED uses non-deterministic forwarding to balance each flow among multiple concurrent routes.
6. **Localized Behavior.** Pure localized algorithms are those in which any action invoked by a node should not affect the system as a whole. In algorithms such as AODV, DSR and TORA, this is not the case. In these protocols a node uses flooding to discover new paths. In sensor networks where thousands of nodes communicate with each other, broadcast storms may result in significant power consumption and possibly a network meltdown. To avoid that, all distributed operations in SPEED are localized to achieve high scalability.
7. **Void Avoidance.** In some scenarios, pure greedy geographic forwarding may fail to find a greedy path to the destination, even when one actually exists. SPEED handles the void the same way as it handles congested areas and guarantees that if there is a greedy route between the source and destination, it will discover it.

Note, while SPEED does not use routing tables, SPEED does utilize location information to carry out routing. Because of this, we assume that each node is location-aware.

4. SPEED Protocol

SPEED maintains a desired delivery speed across sensor networks by both diverting traffic at the networking layer and locally regulating packets sent to the MAC layer. It consists of the following components:

- An API
- A neighbor beacon exchange scheme
- A delay estimation scheme
- The Stateless Non-deterministic Geographic Forwarding algorithm (SNGF)
- A Neighborhood Feedback Loop (NFL)
- Backpressure Rerouting
- Last mile processing

As shown in Figure 1, SNGF is the routing module responsible for choosing the next hop candidate that can support the desired delivery speed. NFL and Backpressure Rerouting are two modules to reduce or divert traffic when congestion occurs, so that SNGF has available candidates to choose from. The last mile process is provided to support the three communication semantics mentioned before. Delay estimation is the mechanism by which a node determines whether or not congestion has occurred. And beacon exchange provides geographic location of the neighbors so that SNGF can do geographic based routing. The details of these components are discussed in the subsequent sections, respectively.
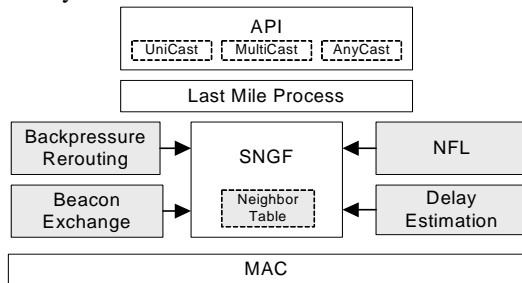

Figure 1. SPEED Protocol

4.1. Application API and Packet Format

The SPEED protocol provides four application-level API calls:

- AreaMulticastSend (position, radius, packet): This service identifies a destination area by its center position and radius. It sends a copy of the packet to every node inside the specified area with a speed above a certain desired value.
- AreaAnyCastSend (position, radius, packet): This service sends a copy of the packet to at least one node inside the specified area with a speed above a certain desired value.
- UnicastSend(Global_ID, packet): In this service the node identified by Global_ID will receive the packet with a speed above a certain desired value.
- SpeedReceive(): this primitive permits nodes to accept packets targeted to them.

Though SPEED is a real-time protocol, we don't use deadline as a parameter in our API. SPEED aims at providing a uniform packet delivery speed across the sensor network, so that the end-to-end delay of a packet is proportional to the distance between the source and destination. With this service, real-time applications can estimate end-to-end delay before making admission decisions. Delay differentiation for different classes of packets is left as future work.

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

- PacketType: the type of communication: Area Multicast, AreaAnyCast or Unicast.
- Global_ID: only used in Unicast communication to identify a destination node.
- Destination Area: Describes a three-dimensional space with a center point and radius in which the packets are destined.
- TTL: Time To Live field is the hop limit used for last mile processing.
- Payload.

4.2. Neighbor Beacon Exchange

Similar to other geographic routing algorithms, every node in SPEED periodically broadcasts a beacon packet to its neighbors. This periodic beaconing is only used for exchanging location information between neighbors. We argue that the beaconing rate can be very low when nodes inside the sensor network are stationary or slow moving. Moreover, piggybacking [6] methods can also be exploited to reduce this beacon overhead.

In addition to periodic beaconing, SPEED uses two types of on-demand beacons, namely a delay estimation beacon and a backpressure beacon, to quickly identify the traffic changes inside the network. The functionality of two beacons will be discussed in section 4.3 and 4.6, respectively. As shown in the evaluation (section 5.4), our on-demand beacon scheme introduces only a small overhead in exchange for a fast response to congestion.

In SPEED, each node keeps a neighbor table to store information passed by the beaconing. Each entry inside the table has the following fields: (NeighborID, Position, SendToDelay, ExpireTime). The ExpireTime is used to timeout this entry. If a neighbor entry is not refreshed after a certain timeout, it will be removed from the neighbor table. SendToDelay is a delay estimation to the neighbor node identified by the NeighborID field. The details of set-

ting this value are discussed in the next section.

4.3. Delay Estimation

We use single hop delay as the metric to approximate the load of a node. We notice that the delays experienced by broadcast packets and unicast packets are quite different due to different handling inside the MAC layer and that unicast packet delay is more appropriate for making routing decisions. In a scarce bandwidth environment, we cannot afford to use probing packets to estimate the single hop delay. Instead we use the data packets passing this node to perform this measurement. Delay is measured at the sender, which timestamps the packet entering the network output queue and calculates the round trip single hop delay for this packet when receiving the ACK. At the receiver side, the duration for processing an ACK is put into the ACK packet. The single-trip time is calculated by subtracting receiver side processing time from the round trip delay experienced by the sender. We compute the current delay estimation by combining the newly measured delay with previous delays via the exponential weighted moving average (EWMA) [8]. Propagation delay is ignored. We argue that this delay estimation is a better metric than average queue size for representing the congestion level of the wireless network, because the shared media nature of the wireless network allows the network to be congested even if queue sizes are small.

4.4. Stateless Non-deterministic Geographic Forwarding (SNGF)

Before elaborating on SGNF, we introduce three definitions:

- The Neighbor Set of Node i: NS_i is the set of nodes that are inside the radio range of node i. Note, we do not assume that the communication radius is a perfect circle. SPEED works with irregular radio patterns.
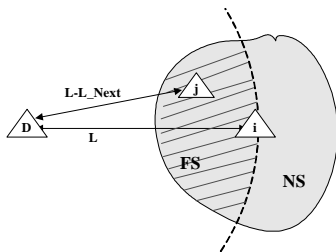

Figure 2. NS and FS definitions

- The Forwarding Candidate Set of Node i: A set of nodes that belong to NS_i and are closer to the destination. Formally, $FS_i (Destination) = \{node \in NS_i \mid L - L\_next > 0\}$ 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 2. We can easily obtain $FS_i (Destination)$ by scanning the NS set of nodes once.

It is worth noticing that the membership of the neighbor set only depends on the radio range, but the membership of the forwarding set also depends on destination area.

- Relay Speed. Relay speed is calculated by dividing the advance in distance from the next hop node j by the estimated delay to forward a packet to node j. Formally,

$$Speed_i^j (Destination) = \frac{L - L\_next}{HopDelay_i^j}.$$

Since in SPEED, nodes keep the Neighbor Set (NS), but don't keep a routing table or flow information, the memory requirements are only proportional to the number of neighbors.

Based on the destination of the packet and the current FS, the Stateless Non-deterministic Geographic Forwarding (SNGF) portion of our protocol routes the packets according to the 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 and a backpressure beacon is issued to upstream nodes to prevent further drops (see 4.7). To reduce the chance of such drops, we deduce a lower bound of node density that can virtually eliminate these drops (appendix A).

2. SPEED divides the neighbor nodes inside $FS_i (Destination)$ into two groups. One group contains the nodes that have relay speeds larger than a certain desired speed $S_{setpoint}$, the other contains the nodes that cannot sustain such desired speed. The $S_{setpoint}$ is a system parameter that depends on the communication capability of the nodes and desired traffic workload a sensor network should support.

3. The forwarding candidate is chosen from the first group, and the neighbor node with highest relay speed 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 Loop (NFL), which is discussed in more detail in section 4.5. 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 speed set point $S_{setpoint}$ and dropping packets must be peformed to reduce the congestion. Though one can consider buffering packets as an alternative to the dropping, however, we argue that under real-time and small memory constrains, dropping is often a better choice.

SNGF provides two nice properties to help meet our design goals. First, since SNGF sends packets to the downstream

node capable of maintaining the desired delivery speed, soft real-time end-to-end delivery is achieved with a theoretical delay bound: *Delay Bound = $L_{e2e}/S_{setpoint}$*, where L_{e2e} is the distance between the source and destination. $S_{setpoint}$ is the uniform speed to be maintained across the sensor network. Second, SNGF can balance traffic and reduce congestion by dispersing packets into a large relay area. This load balancing is valuable in a sensor network where the density of nodes is high and the communication bandwidth is scarce and shared. Load balancing also balances the power consumption inside the sensor networks to prevent some nodes from dying faster than others.

SNGF provides MAC layer adaptation and reduces the congestion by locally dropping (or optionally buffering) packets. This adaptation is good enough to deal with transient overshoot inside the sensor networks. But if such congestion remains for a relatively long time, network layer adaptation is desired to redirect traffic to a less congested area, which is discuss further in section 4.6.

4.5. Neighborhood Feedback Loop (NFL)

The Neighborhood Feedback Loop (NFL) is the key component in maintaining the single hop relay speed. The NFL is an effective approach to maintaining system performance at a desired value. This has been shown in [12], where a low 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 relay speed above a certain value $S_{setpoint}$, a performance goal desired by the system designer.
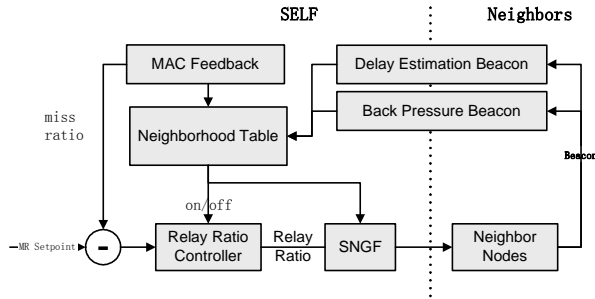

Figure 3. Neighborhood Feedback Loop (NFL)

We deem it a miss when a packet delivered to a certain neighbor node has a relay speed less than $S_{setpoint}$, or if there is a loss due to collision. The percentage of such misses is called this neighbor's miss ratio. The responsibility of the NFL is to force the miss ratios of the neighbors to converge to a set point, namely zero.

As shown in Figure 3, the MAC layer collects miss information and feeds it back to the Relay Ratio controller. The Relay Ratio controller calculates the relay ratio and feeds that into the SNGF where a drop or relay action is made. The Relay Ratio controller currently implemented is a multiple inputs single output (MISO) proportional controller that takes the miss ratios of its neighbors as inputs

and proportionally calculates the relay ratio as the output to the SNGF. Formally it is described by the following formulas.

$$u = 1 - K \frac{\sum e_i}{N} \quad if \quad \forall e_i > 0$$
$$u = 1 \quad if \quad \exists e_i = 0$$

where e_i is the miss ratio of the neighbor i inside the FS set, N is size of the FS set. u is the output (relay ratio) to SNGF. And K is the proportional gain.

It should be noted that the Relay Ratio controller will be activated only when all nodes inside the forwarding set (FS) cannot maintain the desired single hop relay speed $S_{setpoint}$ and a drop is absolutely necessary to maintain the single hop delay. Such a scheme ensures 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.

By reducing the sending rate to the downstream nodes, the neighborhood feedback loop can maintain a single hop relay speed. However, this MAC layer adaptation can't solve the hotspot problem, if the upstream nodes, which are unaware of the congestion, keep sending packets into this area. In this case, backpressure rerouting (network layer adaptation) is necessary to reduce the traffic injected into the congested area.

4.6. Back-Pressure Rerouting

Backpressure re-routing is naturally generated from the collaboration of neighbor feedback loop (NFL) routines as well as the stateless non-deterministic geographic forwarding (SNGF). To be more explicit, we introduce this scheme with an example (Figure 4).
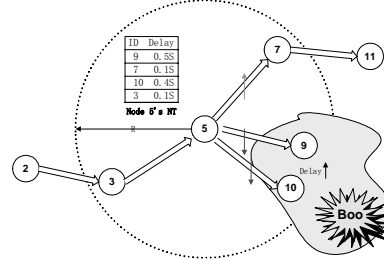

Figure 4. Backpressure rerouting case one

Suppose in the lower-right area, heavy traffic appears, which leads to a lower relay speed in nodes 9 and 10. Through the MAC layer feedback, node 5 will detect that nodes 9 and 10 are congested. Since SNGF will reduce the probability of selecting nodes 9 and 10 as forwarding candidates and route more packets to node 7, it will reduce the congestion around nodes 9 and 10. Since all neighbors of 9 and 10 will react the same way as node 5, eventually nodes 9 and 10 will be able to relay packets above the desired speed.

A more severe case could occur when all the forwarding neighbors of node 5 are also congested as shown in Figure 5.
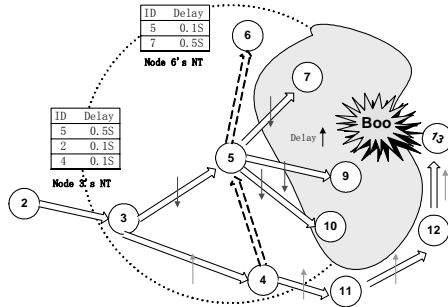

Figure 5. Backpressure rerouting case two

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 in order to reduce the traffic injected into the congested area. At the same time, an on-demand backpressure beacon is issued by node 5 with the following fields.

(ID, Destination, AvgSendToDelay)

AvgSendToDelay is the average SendToDelay of all nodes inside FS_{ID}(Destination). In our example, when the destination is at node 13, AvgSendToDelay is the average delay from node 5 to nodes 7, 9 and 10.

When a neighbor receives the back-pressure beacon from node 5, it determines whether node 5 belongs to its FS(Destination). If node 5 does, this neighbor modifies the SendToDelay for node 5 according to the AvgSendToDelay. For example only node 3 will consider node 5 as a next hop forwarding candidate to the destination where node 13 resides. If node 5 is not in the FS(Destination), then this neighbor ignores the backpressure beacon. This backpressure mechanism can reduce the chance of "false congestion indication", to ensure that traffic from node 4 to node 6 will not be affected by the backpressure beacon.

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 flow to that destination.

Backpressure rerouting is a network layer adaptation used by SPEED to reduce the congestion inside the network. In this case no packet needs to be sacrificed. Network layer adaptation has a higher priority than MAC layer adaptation used by SNGF and NFL. A drop via the feedback loop is only necessary when the situation becomes so congested and there is no alternative to maintaining a single hop speed other than dropping packets.

4.7. Void Avoidance

Greedy geographic based algorithms have many advantages over the traditional MANET routing algorithms for real-time sensor network applications. They do not suffer route discovery delay and tend to choose the shortest path to the destination. Moreover without flooding, they have relatively low control packet overhead. Unfortunately, they also have a serious drawback. In many cases, they may fail to find a path even though one does exist. To overcome this, SPEED deals with a void the same way it deals with congestion. As shown in the Figure 6, if there is no downstream node to relay packets from node 2 to node 5, node 2 will send out a backpressure beacon containing fields: (ID, Destination, ∞). The upstream node 1 that needs node 2 to relay the packets to that destination will set the SendToDelay for node 2 to infinity and stop sending packets to node 2. If node 3 doesn't exist, further backpressure will occur until a new route is found. It should be admitted that our scheme of void avoidance isn't guaranteed to find a path if there is one as in GPSR[6], but it is guaranteed to find a *greedy* path if one exists. To maintain real-time properties, we don't allow backtracking to violate our desired speed setpoint. However, as we can see from the evaluation section 5.6, such a simple scheme can significantly reduce packet loss due to voids in high-density sensor networks.
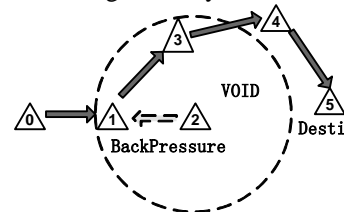

Figure 6. Void avoidance scheme

4.8. Last Mile Process

Since SPEED is targeted at sensor networks where the ID of a sensor node is not important, SPEED only cares about the location where 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. The SNGF module aforementioned controls all previous packet relays.

The last mile process provides two novel services that fit the scenario of sensor networks: Area-multicast and Area-anycast. The area in this case is defined by a center-point (x,y,z) and a radius, in essence a sphere. More complex area definitions can be made without jeopardizing the design of this last mile process.

Nodes can differentiate the packet type by the Packet-Type field mentioned in section 4.1. If it's an anycast packet, the nodes inside the destination area will deliver the packet to the transport layer without relaying it onward. If it's a multicast packet, 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 within the diameter of the destination area and be broadcast within a specified radius. Other nodes inside this 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 the node with a specified 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. The current implementation of the last mile process is relatively simple. More efficient and robust techniques are desired for future research.

5. Experimentation and Evaluation

We simulate SPEED on GloMoSim [15], a scalable discrete-event simulator developed by UCLA. This software provides a high fidelity simulation for wireless communication with detailed propagation, radio and MAC layers. Table 1 describes the detailed setup for our simulator. The communication parameters are mostly chosen in reference to the Berkeley mote specification.

Routing	AODV, DSR, GF, SPEED, SPEED-S, SPEED-T
MAC Layer	802.11 (Simplified DCF)
Radio Layer	RADIO-ACCNOISE
Propagation model	TWO-RAY
Bandwidth	200Kb/s
Payload size	32 Byte
TERRAIN	(200m, 200m)
Node number	100
Node placement	Uniform
Radio Range	40m

Table 1. Simulation settings

In our evaluation, we compare the performance of six different routing algorithms: AODV [10], DSR [5], GF [13], SPEED, SPEED-S, SPEED-T.

GF forwards a packet to the node that makes the most progress toward the destination. SPEED-S and SPEED-T are reduced versions of SPEED. SPEED-S replaces the SNGF with a MAX-SPEED routing algorithm that geographically forwards the packets to nodes that can provide a max single hop relay speed. SPEED-T replaces the SNGF with a MIN-DELAY routing algorithm that geographically forwards packets to nodes that have a minimum single hop delay. Both reduced versions have no backpressure rerouting mechanisms.

In our evaluation, we present the following set of results: 1) end-to-end delay under different congestion levels, 2) miss ratio, 3) control overhead, 4) communication energy consumption, and 5) packet delivery ratio under different node densities. All experiments are repeated 16 times with different random seeds and different random node topologies. We also implement SPEED on the Berkeley motes [4]. The results obtained from this testbed show a load balance feature of SPEED protocol (see section 5.7).

5.1. Sensor Network Traffic Pattern

There are two typical traffic patterns in sensor networks: a base station pattern and a peer-to-peer pattern. The base station pattern is the most representative one inside sensor networks. For example, in surveillance systems, multiple sensors detect and report the location of an intruder to the control center. In tracking systems, a base station issues multiple tracking commands to a group of pursuers. In a different respect, the peer-to-peer pattern is usually used for data aggregation and consensus in a small area where a team of nearby motes interact with each other. The end-to-end delay in the base station pattern is the major part of delay for the sensing-actuation loop, and is therefore, the focus of our evaluation.

5.2. Congestion Avoidance

In a sensor network, where node density is high and bandwidth is scarce, traffic hot spots are easily created. In turn, such hot spots may interfere with real-time guarantees of critical traffic in the network. In SPEED, We apply a combined network and MAC layer congestion control scheme to alleviate this problem.

To test the congestion avoidance capabilities, we use a base station scenario, where 6 nodes, randomly chosen from the left side of the terrain, send periodic data to the base station at the middle of the right side of the terrain. The average hop count between the node and base station is about 8~9 hops. Each node generates 1 CBR flow with a rate of 1 packet/second. To create congestion, at time 80 seconds, we create a flow between two randomly chosen nodes in the middle of the terrain. This flow then disappears at time 150 seconds into the run. This flow introduces a step change into the system, which is an abrupt change that stress-tests SPEED's adaptation capabilities to reveal its transient-state response. In order to evaluate the congestion avoidance capability under different congestion levels, we increase the rate of this flow step by step from 0 to 100 packets/second over several simulations

Figure 7 and Figure 8 plot the end-to-end (E2E) delay for the six different routing algorithms. At each point, we average the E2E delays of all the packets from the 96 flows (16 runs with 6 flows each). The 90% confidence interval is within 2~15% of the mean, which is not plotted for the sake of legibility.

Under the no or light congested situations, Figure 7 and Figure 8 show that all geographic based routing algorithms have short average end-to-end delay in comparison to AODV and DSR. There are several factors accounting for this outcome. First, the route acquisition phase in AODV and DSR leads to significant delays for the first few packets, while geographic based routing doesn't suffer from this. We argue that without an initial delay cost, geographic based routing is more suitable for real-time applications like target 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. Second, the route discovered through

flooding and path reversal has relatively more hops than greedy geographic forwarding. The reason for even higher delay in AODV than DSR is that DSR implementation intensively uses a route cache to reduce route discovery and maintenance cost. As shown in Figure 8, SPEED-T has higher delay than GF, SPEED-S and SPEED, because SPEED-T only uses hop delay to make routing decision and disregards the progress each hop makes, which leads to more hops to the destination in wireless multi-hop networks. Instead, under lightly congested situation, GF, SPEED-S and SPEED tend to forward a packet at each step as close to the destination as possible, thereby reducing the number of hops and the end-to-end delay.
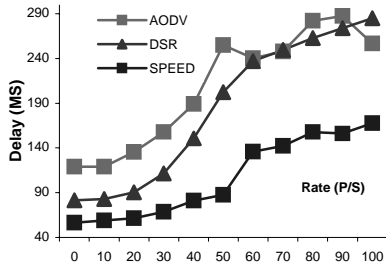
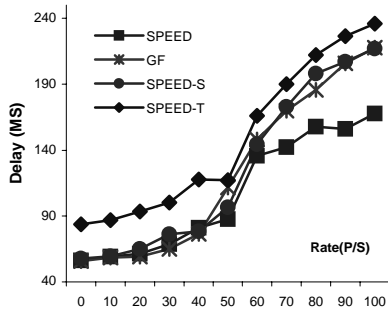Figure 7. E2E Delay Under Different Congestion

Figure 8. E2E Delay Under Different Congestion

Under the heavy congested situations (Figure 7 and Figure 8), each routing algorithm responds differently. SPEED performs best. For example, SPEED reduces the average end-to-end delay by 30%~40% in the face of heavy congestion in comparison to the other algorithms considered. The key reasons for SPEED's better performance are 1) DSR, AODV and GF only respond to severe congestion, which leads to link failures (i.e., when multiple retransmissions fail at the MAC layer). They are insensitive to long delays as long as no link failures occur. 2) DSR, AODV and GF routing decisions are not based on the link delays, and therefore may cause congestion at a particular receiver even though it has long delays. 3) DSR and AODV flood the network to rediscover a new route when the network is already congested. 4) SPEED-T and SPEED-S don't provide traffic adaptation. When all downstream nodes are congested, SPEED-T and SPEED-S cannot reduce or redirect the traffic to uncongested routes. 5) SPEED not only

locally reduces the traffic through a combination of SNGF and Neighborhood Feedback loops in order to maintain the desired speed, but also diverts the traffic into a large area through its backpressure rerouting mechanism. This combination leads to lower end-to-end delay.

5.3. E2E Deadline Miss Ratio

The deadline miss ratio is the most important metric in soft real-time systems. We set the desired delivery speed $S_{setpoint}$ to 1km/s, which leads to an end-to-end deadline of 200 milliseconds. In the simulation, some packets are lost due to congestion or forced-drops. We also consider this situation as a deadline miss. The results shown in Figure 9 and Figure 10 are the summary of 16 randomized runs.
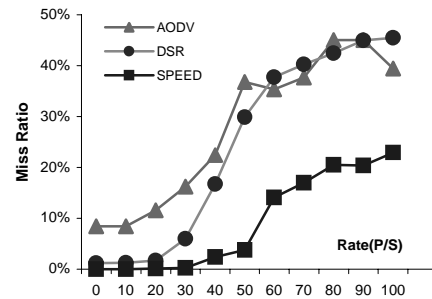

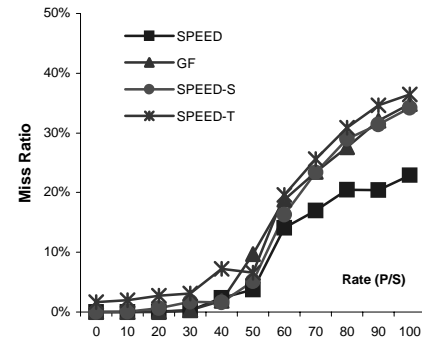Figure 9. MissRatio Under Different Congestion

Figure 10. MissRatio Under Different Congestion

AODV and DSR don't perform well in the face of congestion because both algorithms flood the network in order to discover a new path when congestion leads to link failure. This flooding just serves to increase the congestion. GF only switches the route when there are link failures caused by heavy congestion. The routing decision is based solely on distance and does not consider delay. SPEED-T only considers the single hop delay and doesn't take distance (progress) into account, which leads to a longer route. SPEED-S provides no adaptation to the congestion and cannot prevent packets from entering the congestion area. Only SPEED tries to maintain a desired delivery speed through MAC and network layer adaptations, and therefore has a much less miss ratio than other algorithms. Due to its transient behavior, SPEED still has about a 20% miss ratio when the network is heavily congested. Future work is

needed to reduce the convergence time in order to improve the performance.

Comparing Figure 9 and Figure 10, we argue that purely localized algorithms without flooding outperform other algorithms when traffic congestion increases. Generally, the less state information a routing algorithm depends on, the more robust it is in the face of packet loss and congestion.

5.4. Control Packet Comparison

Except for AODV, all other 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 initiates route discovery (flooding) whenever a link breaks due to congestion, it requires a large number of control packets. DSR uses a route cache extensively, so it can do route discovery and maintenance with a much lower cost than AODV. The only control packets used in GF, SPEED-S and SPEED-T (Figure 11) are periodic beacons, whose number is constant at 750 under different congestion levels. In addition to periodic beacons, SPEED uses two types of on-demand beacons to notify neighbors of the congestion. This costs SPEED more control packets than the other three geographic based routing algorithms (Figure 11).
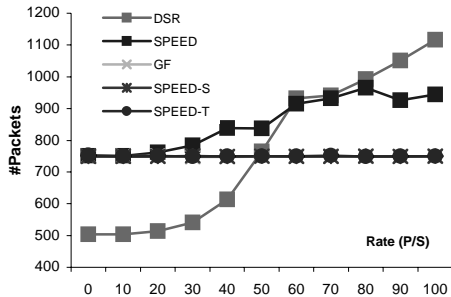

Figure 11. Control packet overhead comparison

5.5. Energy Consumption

Under energy constraints, it is vital for sensor nodes to minimize energy consumption in radio communication to extend the lifetime of sensor networks. From the results shown in Figure 12, we argue that geographic based routing tends to reduce the number of hops in the route, thus reducing the energy consumed for transmission. AODV performs the worst as a consequence of sending out many control packets during congestion. DSR has larger average hop counts and more control packets than other geographic base routing algorithms. SPEED-T only takes delay into account, which leads to longer routes. Figure 12 shows that SPEED has nearly the same power consumption as GF and SPEED-S when the network is not congested. Under such situations, SPEED tends to choose the shortest route and does not require any on-demand beacons. Under heavy congestion, SPEED has slightly higher energy consumption than GF

and SPEED-S, mainly because SPEED delivers more packets to the destination than the other protocols when heavily congested.
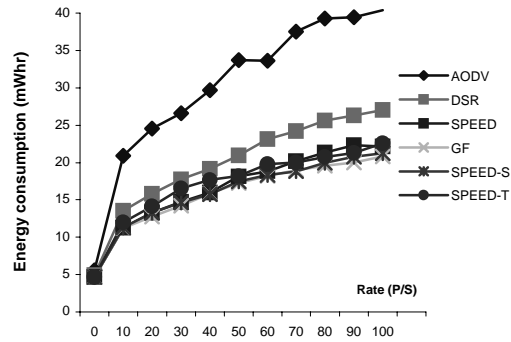

Figure 12. Energy Consumption for transmission
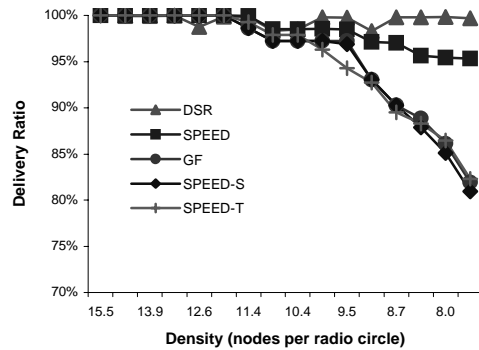
5.6. Void Avoidance

Figure 13. Deliver ratio under different density

This experiment tries to evaluate the end-to-end delivery ratio of all routing algorithms under different node densities. To eliminate packet loss due to the congestion, we only use four flows with a rate of 0.5 packets/second, these flows go from the left side of the terrain to the base station at the right side of the terrain. To change the density of the network, instead of increasing the number of nodes in the terrain, we keep the number of nodes constant at 100, and increase the side length of the square terrain in steps of 50 meters. It is no surprise that DSR performs best in the delivery ratio since it is a flooding based route discovery algorithm. Theoretically, DSR should have 100% delivery ratio (Figure 13) as long as the network is not partitioned. All other geographic based algorithms have 100% delivery ratio when the network has high density (>12 nodes / per radio range). However, when the network density is reduced below 9 nodes/ per radio circle, GF, SPEED-S and SPEED-T degrade performance rapidly. Only SPEED can manage to deliver 95% of its packets to the destination. However, SPEED drops 5% of its packets, because those packets need backtracking in order reach the destination. If backtracking, those packets would have a negative delivery speed, which

is not allowed by SPEED for the sake of maintaining the real-time properties. It should be pointed out that GPSR[6], another well known geographic based routing algorithm, permits backtracking and can achieve 100% delivery rate as long as the network is not partitioned.

5.7. Implementation on Motes

We have implemented the SPEED protocol on Berkeley motes platform with a code size of 6036 bytes (code is available at [3]). Three applications including data placement, target tracking and CBR are built on top of SPEED. Due to space limitation, we only present partial results here. In the experiment, we use 25 motes to form a 5 by 5 grid. To evaluate the load balance capability of the SPEED, we send a CBR flow from node 24 to node 0 which is the base station. We collect the number of packets relayed by intermediate motes (1~23) and compare this with the result obtained from GF protocol which we also implemented on the motes.

GF tends to relay packets via a fixed route which leads to unbalance traffic, for example, in Figure 14, node 14 sends out 98 packets while node 13 doesn't sent out any packets. SPEED uses non-deterministic forwarding, which can balance energy consumption. We argue that in sensor networks, balanced energy consumption can prevent some nodes from dying faster than others, therefore increasing the network lifetime.
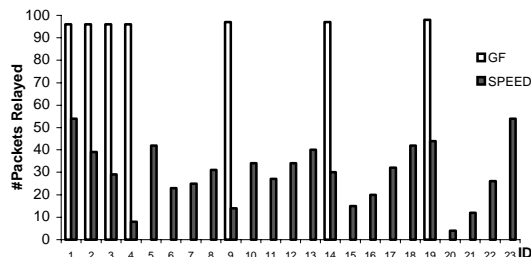

Figure 14. Traffic Balance

6. Conclusion

Many excellent protocols have been developed for ad hoc networks. However, 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 maintains a desired delivery speed across the network through a novel combination of feedback control and non-deterministic QoS-aware geographic forwarding. This combination of MAC and network layer adaptation improves the end-to-end delay and provides good response to congestion and voids. Our simulations on GloMoSim and implementation on Berkeley motes demonstrate SPEED's improved performance compared to DSR, AODV, GF, SPEED-S and SPEED-T. SPEED is a new

protocol that meets the requirements of sensor networks in real-time situations.

7. Acknowledgment

References

[1] G. S. Ahn, A. T. Campbell, A. Veres and L.H. Sun. SWAN: Service Differentiation in Stateless Wireless Ad Hoc Networks, In *Proc. IEEE INFOCOM'2002*, June 2002.

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

[3] T. He, L. Gu, B.Blum, Jun Xie. Nest Project Source Code http://sourceforge.net/projects/vert/

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

[5] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Mobile Computing*, Chapter 5, pages 153-181, Kluwer Academic Publishers, 1996.

[6] B. Karp and H. T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *IEEE MobiCom*, August 2000.

[7] Y.B. Ko and N. H. Vaidya. Location-Aided Routing(LAR) in Mobile Ad Hoc Networks. In *IEEE MobiCom 1998*, October 1998.

[8] J. F. Kurose, K. W. Ross. Computer Networking A Top-Down Approach Featuring the internet. ISBN 0-201-47711-4 Addison Wesley Longman Inc.

[9] C. Lu, B. M. Blum, T. F. Abdelzaher, J. A. Stankovic, and T. He. RAP: A Real-Time Communication Architecture for Large-Scale Wireless Sensor Networks*, In IEEE RTAS 2002*, September 2002.

[10] C. E. Perkins and E. M. Royer. Ad-hoc On Demand Distance Vector Routing. In *WMCSA'99*, February 1999.

[11] C. E. Perkins and P. Bhagwat. Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for Mobile Computers, in *SIGCOMM Symposium on Communications Architectures and Protocols*, pp. 212-225, September. 1994.

[12] J. A. Stankovic, T. He, T. F. Abdelzaher, M. Marley, G. Tao, S. Son, and C. Lu. Feedback Control Scheduling in Distributed Systems*, IEEE RTSS*, December 2001.

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

[14] A. Woo and D. Culler. A Transmission Control Scheme for Media Access in Sensor Networks, In *IEEE MobiCOM 2001*, July 2001.

[15] X. 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.