MMSPEED: Multipath Multi-SPEED Protocol for QoS Guarantee of Reliability and Timeliness in Wireless Sensor Networks

Emad Felemban, *Student Member*, *IEEE*, Chang-Gun Lee, *Member*, *IEEE*, and Eylem Ekici, *Member*, *IEEE*

Abstract—In this paper, we present a novel packet delivery mechanism called *Multi-Path and Multi-SPEED Routing Protocol (MMSPEED)* for probabilistic QoS guarantee in wireless sensor networks. The QoS provisioning is performed in two quality domains, namely, timeliness and reliability. Multiple QoS levels are provided in the timeliness domain by guaranteeing multiple packet delivery speed options. In the reliability domain, various reliability requirements are supported by probabilistic multipath forwarding. These mechanisms for QoS provisioning are realized in a localized way without global network information by employing localized geographic packet forwarding augmented with *dynamic compensation*, which compensates for local decision inaccuracies as a packet travels towards its destination. This way, MMSPEED can guarantee end-to-end requirements in a localized way, which is desirable for scalability and adaptability to large scale dynamic sensor networks. Simulation results show that MMSPEED provides QoS differentiation in both reliability and timeliness domains and, as a result, significantly improves the effective capacity of a sensor network in terms of number of flows that meet both reliability and timeliness requirements up to 50 percent (12 flows versus 18 flows).

Index Terms—System design, simulations, sensor networks, service differentiation, QoS, real-time, reliability, localized routing protocol.

1 Introduction

Wireless sensor networks can be used for many mission-critical applications such as target tracking in battlefields and emergency response. In these applications, reliable and timely delivery of sensory data plays a crucial role for the success of the mission. Specifically, the abovementioned sensor network applications share the following characteristics:

- Diverse *Real-Time* Requirements: Some sensory data reflects the physical status of dynamically changing environment such as positions of moving targets and temperatures of forest areas. Such sensory data is valid only for a limited time duration and, hence, needs to be delivered within a time deadline for real-time applications. More importantly, different sensory data has different deadlines depending on the dynamics of the sensed environment. For example, location information of a fast moving target has shorter deadline than that of a slow moving target.
- Diverse *Reliability* Requirements: Depending on its contents, sensory data may have different reliability requirements. For example, in a forest monitoring application, the temperature information in the normal temperature range can be delivered to the control center tolerating a certain percentage of loss. On the other hand, the sensor data reflecting a high temperature should be delivered to the control center with a very high probability since it can be a sign of fire.
- Mixture of periodic and aperiodic data: Some sensory data are created aperiodically by detection of critical events at unpredictable times. In addition, there are other types of sensory data created periodically for continuous real-time monitoring of environmental status.

QoS provisioning for the above diverse flows is a challenging problem due to the following characteristics of sensor networks:

- dynamic topology changes due to node mobility, failure, and addition;
- large scale with thousands of densely placed nodes; and
- less reliable nature due to noisy wireless links.[1]

Existing QoS provisioning protocols [2], [3], [4], [5] in wireless ad hoc networks are based on the end-to-end path discovery, resource reservation along the discovered path, and path recovery in case of topological changes. However, such approaches are not suitable for sensor network applications with above characteristics for many reasons. First, the path discovery latency is not acceptable for urgent aperiodic packets. Furthermore, it is not practical to reserve resources for the unpredictable aperiodic packets. Even for periodic continuous flows, the end-to-end path based mechanisms are problematic in dynamic sensor networks

• *The authors are with the Department of Electrical and Computer Engineering, The Ohio State University, 2015 Neil Avenue, Columbus, OH 43210. E-mail: {felembe, cglee, ekici}@ece.osu.edu.*

Manuscript received 4 May 2005; revised 30 Aug. 2005; accepted 15 Sept. 2005; published online 17 Apr. 2006.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-0128-0505.

1. A recent study [1] reports that 20 percent of neighbor nodes in a radio communication range suffer more than 10 percent of packet loss.

since service disruption during the path recovery is not acceptable in mission critical applications. Finally, the reservation-based approaches are not scalable due to huge overhead of path discovery and recovery in large scale sensor networks.

Other QoS studies in sensor networks like [6], [7] focus on only one QoS domain, either timeliness or reliability. They are also limited in differentiating services for traffic flows with different levels of timeliness and reliability requirements. Another study [8] can guarantee the different real-time requirements by realizing EDF packet scheduling in a decentralized way. However, it is based on the assumption that most traffic is periodic and all periods are known a priori, which is not the case for many sensor network applications. Also, the solution presented in [8] fails to adapt to dynamics of sensor networks.

In this paper, a novel packet delivery mechanism for QoS provisioning called *Multi-Path and Multi-SPEED Routing Protocol (MMSPEED)* is proposed, which spans over network layer and medium access control (MAC) layer.[2] Our major goal is to provide QoS differentiation in two quality domains, namely, *timeliness* and *reliability*, so that packets can choose the most proper combination of service options depending on their timeliness and reliability requirements. The power consumption problem is beyond the scope of this paper since we target short-living sensor network applications whose mission duration lasts only for few hours or one day at most and, hence, QoS support for the mission duration is more important than prolonging the network lifetime. For the service differentiation in the timeliness domain, the proposed mechanism provides *multiple network-wide speed options* extending the idea of single network-wide speed guarantee in [9]. For the service differentiation in the reliability domain, we exploit the inherent redundancy of dense sensor networks by realizing *probabilistic multipath forwarding* depending on packet's reliability requirement.

Another important property of MMSPEED is end-to-end QoS provisioning with local decisions at each intermediate node without end-to-end path discovery and maintenance. This property is important for scalability to large sensor networks, self-adaptability to network dynamics, and appropriateness to both aperiodic and periodic traffic flows. For this, MMSPEED realizes the above QoS differentiation based on localized geographic forwarding using only immediate neighbor information. One challenge is to ensure that localized forwarding decisions result in end-to-end QoS provisioning in global sense. To handle this problem, we propose the notion of *dynamic compensation*, which compensates for inaccuracy of local decisions in a global sense as a packet progresses towards its destination. As a result, packets can meet their end-to-end requirements with a high probability even if packet delivery decisions are made locally.

The rest of this paper is organized as follows: Section 2 summarizes the related work. Section 3 presents the proposed routing protocol. Section 4 describes our add-on features of MAC protocol to support the routing protocol. Section 5 presents an optimal protocol configuration by trading-off timeliness and reliability. Section 6 discusses the performance evaluation of the proposed protocols. Finally, Section 7 concludes the paper.

2. Regarding the MAC layer, we mostly rely on IEEE 802.11e standard and present only necessary add-on features.

2 Related Work

In literature, several QoS provisioning protocols have been proposed for wireless ad hoc networks [2], [3], [4], [5]. However, they are based on end-to-end path discovery and resource reservation, which renders their application impractical for large scale dynamic sensor networks.

Recent QoS studies in sensor networks focus on only one QoS domain, either reliability (e.g., AFS [7] and ReInforM [10]) or timeliness (e.g., RAP [6], Implicit EDF [8], and SPEED [9]). AFS [7] and ReInforM [10] are two examples that leverage path redundancy in wireless sensor networks for service differentiation in the reliability domain. However, both protocols require the global knowledge of the network topology and also they are limited in differentiating services in the timeliness domain.

RAP [6] provides service differentiation in the timeliness domain by velocity-monotonic classification of packets. Based on packet's deadline and destination, its required velocity is calculated and its priority is determined in the velocity-monotonic order so that a high velocity packet can be delivered earlier than a low velocity one. However, it is best-effort service differentiation without any guarantee in the end-to-end sense. Implicit EDF [8] can provide hard real-time guarantee based on decentralized EDF packet scheduling. However, it works only when most traffic is periodic and all periods are known a priori, which is not the case for many sensor network applications. Also, it is not adaptive to dynamics of sensor networks.

SPEED [9] protocol is designed to provide soft end-to-end deadline guarantees for real-time packets in sensor networks. It uses a geographic forwarding mechanism such that each packet can be routed without global topology information. Thus, it can scale well to a large scale sensor network. More importantly, it ensures a network wide speed of packet delivery for real-time guarantee. For this, each node maintains neighbor node information such as geographic distance and average delay to each neighbor. Using the distance and delay, each node evaluates the packet progress speed of each neighbor node and forwards a packet to a node whose progress speed is higher than the prespecified lower-bound speed $SetSpeed$. If each node can find a neighbor that can progress a packet with a speed higher than $SetSpeed$, $SetSpeed$ can be guaranteed network-widely. However, if workload is too heavy, it is not always possible. When a node cannot find any neighbor node whose speed is higher than $SetSpeed$, it probabilistically drops packets to regulate the workload such that at least one neighbor node with a speed higher than $SetSpeed$ exists at all times. At the same time, the node sends a back-pressure packet to the previous nodes to prevent them from forwarding any further packets through this congested area. This SPEED protocol has many nice features: 1) $SetSpeed$ is uniformly guaranteed all over the network and thus we can predict if the end-to-end deadlines of packet can be met and 2) every mechanism works in a localized way and, hence,

SPEED is quite scalable. However, the SPEED protocol provides only one network-wide speed, which is not suitable for differentiating various traffic with different deadlines. In addition, it is limited to provide any guarantee in the reliability domain.

Mobicast [11] aims at reliable and just-in-time delivery of alert packets to all sensor nodes in the moving delivery zone. This service is useful for waking up sensors ahead in the target trajectory being tracked. However, it assumes reliable and time-bounded transmission between every pair of sensor nodes and uses all nodes in a quite large forwarding zone to forward packets.

None of the existing protocols can achieve all the following goals at the same time:

- Service differentiation and guarantee in both timeliness and reliability,
- Localized packet delivery without global topology information, and
- Overcoming less-reliable and unbounded transmission over wireless links.

3 MMSPEED: Multi-Path and Multi-SPEED Routing Protocol

The proposed routing protocol is designed with two important goals:

- localized packet routing decision without global network state update or a priori path setup, and
- providing differentiated QoS options in timeliness and reliability domains.

For the localized packet routing without end-to-end path setup and maintenance, we adopt the geographic routing mechanism based on location awareness. First, we assume that the packet destination is specified by a geographic location rather than node ID as justified in the SPEED paper [9]. Also, each sensor node is assumed to be aware of its geographical location using GPS [12] or distributed location services [13], [14]. The location information can be exchanged with immediate neighbors with "periodic location update packets." Thus, each node is aware of its immediate neighbors within its radio range and their locations. Using the neighbor locations, each node can locally make a per-packet routing decision such that packets progress geographically towards their final destinations. If each node relays the packet to a neighbor closer to the destination area, the packet can eventually be delivered to the destination without global topology information. Many recent protocols [15], [16], [12], [9] are also employing such geographic routing mechanisms. The localized geographic routing has the following three advantages in sensor networks:

- scalability to a very large and dense sensor networks,
- no path setup and recovery latency—suitable for both critical aperiodic and periodic packets, and
- Per-packet path discovery resulting in self adaptation to network dynamics.

Our goal is to provide guaranteed packet delivery services in both timeliness and reliability domains while preserving the benefits of localized geographic routing.

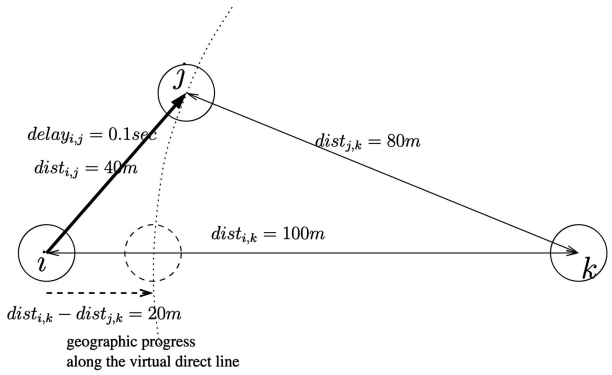


Fig. 1. Progress speed from node i to node j toward destination k.

3.1 Differentiated QoS Options in the Timeliness Domain

For on-time delivery of packets with different end-to-end deadlines, MMSPEED provides multiple delivery speed options that are guaranteed network-widely. For this, we borrow the idea of SPEED protocol [9] which can guarantee a single network-wide speed.

Consider two immediate neighbor nodes i and j in Fig. 1. The geographical distances from node i and node j to the final destination k are $dist_{i,k} = 100m$ and $dist_{j,k} = 80m$, respectively. Suppose that node i forwards a packet to node j with delay (including queueing, processing, and MAC collision resolution) of $delay_{i,j} = 0.1sec$. This forwarding makes $dist_{i,k} - dist_{j,k} = 20m$ geographic progress toward the final destination k along the virtual direct line from node i to destination k. Thus, the progress speed $Speed^k_{i,j}$ from node i to node j toward the final destination k is $(dist_{i,k} - dist_{j,k})/delay_{i,j} = 20m/0.1sec = 200m/sec$. If every node i in the entire network can relay a packet to a neighbor node j whose progress speed toward destination k, i.e., $Speed^k_{i,j} = (dist_{i,k} - dist_{j,k})/delay_{i,j}$, is higher than the prespecified lower-bound speed $SetSpeed$, then the $SetSpeed$ can be uniformly guaranteed all over the network. If such network-wide guarantee of $SetSpeed$ is possible, the end-to-end packet delivery delay from any source s to any destination d can be bounded by $dist_{s,d}/SetSpeed$.

For this purpose, in SPEED protocol, each node i maintains delay estimation $delay_{i,j}$ to each neighbor j, calculates $Speed^k_{i,j} = (dist_{i,k} - dist_{j,k})/delay_{i,j}$, and forwards a packet to a neighbor j whose progress speed $Speed^k_{i,j}$ is higher than $SetSpeed$. However, nodes in a congested area may not be able to find any node with progress speed higher than $SetSpeed$. Those nodes start reducing workload by probabilistically dropping packets in order to retain at least one forwarding node whose progress speed is higher than $SetSpeed$. This approach compromises reliability for assuring network-wide uniform speed $SetSpeed$ with a high probability. Along with packet dropping, nodes also issue so-called "back-pressure packets" to reduce the incoming packet traffic from other neighboring nodes. This back-pressure mechanism can also solve the void area problem, where routes may not find any neighbors that are closer to the destination than themselves.

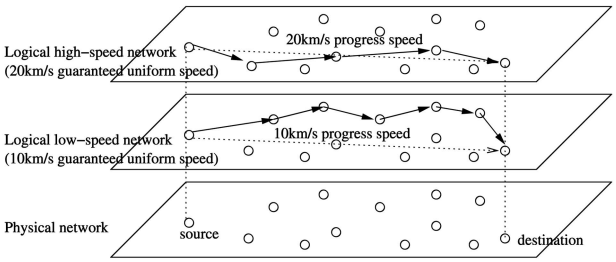


Fig. 2. Virtual overlay of multiple speed layers.

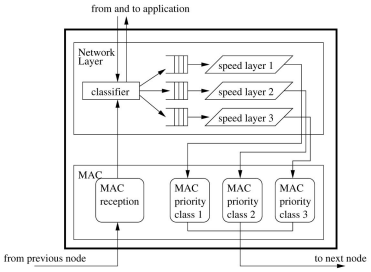


Fig. 3. Protocol structure of each sensor node.

By replicating the single network-wide speed guarantee mechanism, our protocol provides multiple layers of network-wide speed guarantees. From the network wide point of view, our protocol can be conceptually understood as a virtual overlay of multiple SPEED layers on top of a single physical network as depicted in Fig. 2. For this, each node has the protocol structure as in Fig. 3. Each speed layer l independently runs the above SPEED mechanism to guarantee the corresponding $SetSpeed_l$. We will explain how to determine $SetSpeed_l$ in Section 5. For now, we assume that $SetSpeed_l$ is given a priori for each speed layer l. For this virtual layering, our protocol employs two important notions:

- Virtual isolation among the speed layers,
- Dynamic compensation of local decisions.

The virtual isolation of the speed layers aims to minimize the effects of lower speed packets on the delays experienced by the higher speed packets. Virtual isolation is accomplished by classifying incoming packets according to their speed classes and placing them into appropriate priority queues as shown in Fig. 3. The packets in the highest speed queue is served in FCFS discipline, followed by the next highest speed queue, and so on. This packet processing strategy prevents a packet of higher speed layer from being delayed by lower speed packets in the same node. Even if this local prioritization is possible in the network layer, prioritized transmission among neighbors is not possible due to the randomness of CSMA/CA mechanism used in a normal MAC protocol. For the prioritized transmission among neighbors, a special support from MAC layer is needed. This issue will be discussed in Section 4.

The dynamic compensation is needed to adjust the local decisions to meet the end-to-end deadline. Specifically, the classifier of the source node s selects the most proper speed for a packet x based on the distance to final destination d, i.e., $dist_{s,d}(x)$ and end-to-end deadline $deadline(x)$. The minimum required speed level $ReqSpeed(x)$ for a packet x to meet the end-to-end deadline is calculated as

$$ReqSpeed(x) = \frac{dist_{s,d}(x)}{deadline(x)}. \quad (1)$$

Thus, the classifier of the source node picks the most proper speed layer l such that

$$SetSpeed_l = min_{j=1}^{L}\{SetSpeed_j | SetSpeed_j \geq ReqSpeed(x)\}, \quad (2)$$

where L is the number of speed options. Then, the corresponding speed layer module chooses a neighbor node i whose progress speed estimation $Speed_{s,i}^{d} = (dist_{s,d} - dist_{i,d})/delay_{s,i}$ is higher than $SetSpeed_l$.

However, after the packet travels several hops towards the destination d, an intermediate node m may notice that the packet has traveled slowly so far due to longer delays than the original estimations. This can be noticed by comparing the expected latency to the destination using the current speed, i.e., $\frac{dist_{m,d}(x)}{SetSpeed_l}$, and the remaining time to deadline. However, determining the remaining time to deadline at each intermediate node is not trivial due to the lack of globally synchronized clocks. To handle this problem, we measure the *elapsed time* at each node m and piggyback the elapsed time to the packet so that the following node m' can determine the remaining time to deadline without using a globally synchronized clock. For this, when a node m receives the last bit of a packet x, its MAC layer tags $t^{arrival}$ to the packet. This packet is processed by the network layer and forwarded to the chosen forwarding node m' via MAC layer. Note that the MAC layer of m spends some time to capture the channel using RTC/CTS handshake and may transmit the packet several times until receiving ACK from m'. For m to piggyback the accurate elapsed time, the MAC layer updates the field of elapsed time $t^{elapsed}$ every time just before it actually transmits the packet x to the physical link as follows

$$t^{elapsed} = t^{departure} + t^{transDelay} - t^{arrival}, \quad (3)$$

where $t^{departure}$ is the time when node m transmits the first bit of packet x to the physical link and $t^{transDelay}$ is the transmission delay of packet x which can be computed using the transmission rate and the packet length. Thus, once node m' successfully receives the packet, the packet contains the correct measurement of the elapsed time at node m. Now, node m' can update the remaining time to deadline as follows:

$$deadline(x) = deadline(x) - t^{elapsed} - t^{propDelay}, \quad (4)$$

where $t^{propDelay}$ is the propagation delay between two neighbor nodes, which is negligibly small.

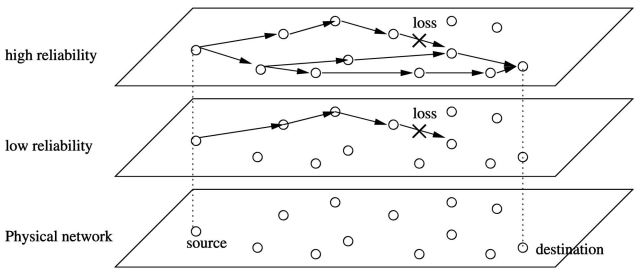


Fig. 4. Service differentiation in reliability domain.

Using this new remaining time to deadline, i.e., $deadline(x)$, if the intermediate node m' notices that the current speed $SetSpeed_l$ is insufficient, i.e., $\frac{dist_{m',d}}{SetSpeed_l} > deadline(x)$, node m' can boost the speed level using the following formula:

$$ReqSpeed(x) = \frac{dist_{m',d}}{deadline(x)},$$

$$SetSpeed_l = \min_{j=1}^{L}\{SetSpeed_j | SetSpeed_j \geq ReqSpeed(x)\}.$$

By implementing this speed level compensation in the classifier in Fig. 3, inaccuracies of localized decisions can be compensated globally as the packet travels.

Thanks to the network-wide speed options together with dynamic compensation, we can claim that once a packet reaches its destination, it is likely that the packet meets its end-to-end deadline. However, not all packets are guaranteed to reach their destinations since we are compromising the reachability by intentionally dropping packets to guarantee the network-wide speed options. To assure a certain level of reachability, we propose another mechanism in the reliability domain as described in the next section.

3.2 QoS Differentiation in the Reliability Domain

In a dense sensor network, there exist multiple redundant paths to the final destination [17], [18], [10] even though they may not be the shortest paths. A nonshortest path is acceptable as long as it can deliver a packet within the end-to-end deadline. Furthermore, utilizing possibly longer alternative paths is sometimes preferable for load balancing and avoiding hot spots on the shortest paths. Our MMSPEED protocol exploits such inherent redundancies to probabilistically guarantee the required end-to-end reliability level (end-to-end reaching probability) of a packet. The probability that the packet reaches its final destination grows as the number of paths used to deliver a packet increases, despite of packet drops, node failures, and errors on wireless links. Thus, by controlling the number of forwarding paths depending on the required reliability level, we can provide the service differentiation in the reliability domain. This idea is depicted in Fig. 4 where a low reliability packet is delivered using a single path while a high reliability packet is delivered over multiple paths.

The challenging task is to devise local decision mechanisms to compute and identify forwarding paths to meet packet's end-to-end reachability requirement. To address this problem, we combine 1) multipath forwarding based on local estimation and 2) dynamic compensation. Each node locally determines multiple forwarding nodes to meet the required reaching probability based on local error estimations and geographic hop distances to immediate neighbors. More specifically, each node i can maintain the recent average of packet loss percentage $e_{i,j}$ to each immediate neighbor node j. The packet loss includes both intentional packet drops for congestion control and errors on the wireless channel. The calculation of packet loss percentage is also supported by MAC layer loss estimation as described in Section 4. Using $e_{i,j}$ as an estimate of packet loss probability, node i can locally estimate the end-to-end reachability of a packet from node i to the final destination d via a neighbor node j as follows:

$$RP_{i,j}^{d} = (1 - e_{i,j})(1 - e_{i,j})^{\lceil dist_{j,d}/dist_{i,j} \rceil}, \quad (5)$$

where $\lceil dist_{j,d}/dist_{i,j} \rceil$ is hop count estimation from node j to the final destination d. Note that this local estimation equation is based on two assumptions: 1) packet loss rate in each of the following hops will be similar to the local loss rate of the current hop and 2) for each following hop, the geographic progress to the destination will be similar to the current progress.

From the end-to-end reachability estimation $RP_{i,j}^{d}$ via a single neighbor node j, we can determine the number of forwarding paths to satisfy the end-to-end reachability requirement P^{req} of a packet. More specifically, we initially set the total reaching probability TRP to zero. If we use one more forwarding path via node j, the TRP can be updated as follows:

$$TRP = 1 - (1 - TRP)(1 - RP_{i,j}^{d}). \quad (6)$$

In the equation, $(1 - TRP)$ is the probability that none of the current paths can successfully deliver the packet to the destination and $(1 - RP_{i,j}^{d})$ is the probability that the one additional path via node j will fail to deliver the packet to the destination. Thus, $(1 - TRP)(1 - RP_{i,j}^{d})$ is the probability that all paths including new one will fail to deliver the packet. Therefore, $1 - (1 - TRP)(1 - RP_{i,j}^{d})$ is the probability that at least one path will successfully deliver the packet to the destination. Using this TRP estimation,[3] we can include more neighbor nodes for packet forwarding until TRP becomes larger than P^{req}. Once we determine the set of required forwarding nodes, the packet is delivered to them using MAC multicast service described in Section 4.

However, local decisions on multiple forwarding node selection may turn out to be incorrect in the following nodes because local estimations are used to model the remaining part of the network about which the local node does not have any information. To address this problem, we use dynamic compensation in the reliability domain. The dynamic compensation can be explained with an example in Fig. 5. Consider a source sensor node s that detects an event that needs to be reported to the control center d with reachability $P^{req} = 80\%$. Suppose that the source node s determines to forward this packet to two immediate neighbors j_1 and j_2 based on its local estimation of $RP_{s,j_1}^{d} = 70\%$ and

3. For the stability of TRP estimation, we assume that the node mobility is much lower than packet information collection rate—movement occurs only every few seconds, which is true for most real sensor networks.

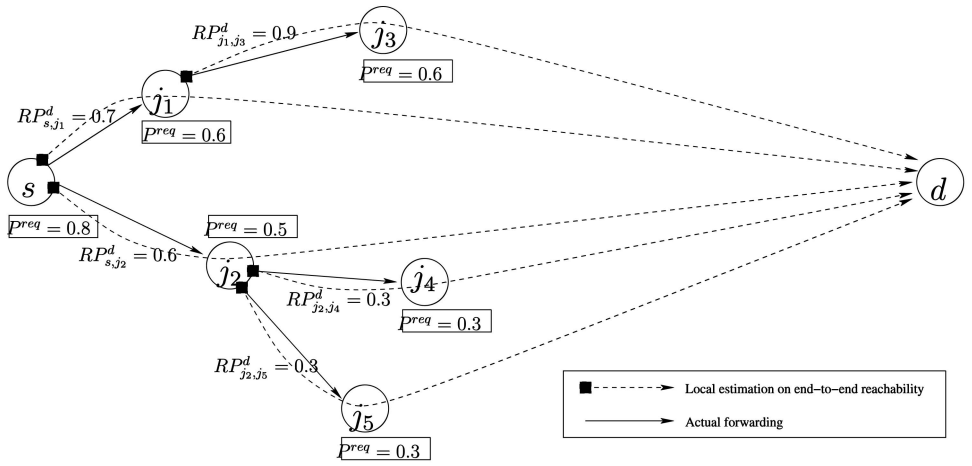


Fig. 5. Multipath forwarding and dynamic compensation.

$RP^d_{s,j_2} = 60\%$. Remember that total reaching probability TRP via node j_1 and j_2 is given as

$$TRP = 1 - (1 - RP^d_{s,j_1})(1 - RP^d_{s,j_2}) = 1 - (1 - 0.7)(1 - 0.6) = 0.88,$$

which is higher than the reachability requirement $P^{req} = 80\%$. When transmitting the packet to node j_1 and j_2, new P^{req} values are assigned for each recipient. For example, recipients j_1 and j_2 may be assigned with $P^{req} = 0.6$ and $P^{req} = 0.5$, respectively, to just meet the condition that $TRP = 1 - (1 - 0.6)(1 - 0.5) = 0.8$.

When nodes j_1 and j_2 receive their copies with assigned $P^{req} = 0.6$ and $P^{req} = 0.5$, respectively, they make local forwarding decision to meet P^{req} as before but using their own estimations. For example, node j_1 can find a forwarding neighbor j_3 with $RP^d_{j_1,j_3} = 0.9$, and thus the assigned requirement $P^{req} = 0.6$ can be met with this single forwarding path. Thus, the packet is forwarded from j_1 to j_3 without change of $P^{req} = 0.6$. On the other hand, node j_2 finds that it needs two forwarding nodes j_4 and j_5 with $RP^d_{j_2,j_4} = 0.3$ and $RP^d_{j_2,j_5} = 0.3$ since its local loss rate estimation is worse than the original one made in the source node s. In this case, j_2 delivers the packet to j_4 and j_5 with adjusted values of $P^{req} = 0.3$ and $P^{req} = 0.3$, respectively, (the total reaching probability through these two nodes is $1 - (1 - 0.3)(1 - 0.3) = 0.51$) to meet the requirement $P^{req} = 0.5$. This way, each following node dynamically compensates the previous wrong decision as the packet travels to the final destination.

Along with this hop-by-hop dynamic compensation, we also employ *reliability back-pressure mechanism* to remedy the problem of local decision in a more global scope. Since the sending node i assigns P^{req} based on its local estimation, it is possible that the receiving node j cannot satisfy the assigned P^{req} even with the hop-by-hop dynamic compensation using all possible forwarding nodes. If this over-expectation is detected by node j, it issues reliability back-pressure packet to reduce the reliability expectation of previous nodes. Specifically, the receiving node j detecting over-expectation issues a back-pressure packet with its maximum possible TRP that can be calculated by (6) adding RPs of all forwarding nodes. If the previous sender node i receives this back-pressure packet, it will use TRP as the maximum value of P^{req} that can be assigned to node j for delivering future packets. In an extreme case, this back-pressure can propagate to the original source so that P^{req} assignment can be made correctly from the beginning. This way, we can remedy the incorrectness of local decision more globally when necessary. A node that receives a back-pressure packet starts a timer called $T_{backpressure}$ to return to the normal operation expecting that the conditions that caused the back-pressure packets have been resolved. The duration of $T_{backpressure}$ should match the average duration of abnormal conditions. Since the average duration differs depending on applications and operation scenarios, we have to rely on previous experiences and statistics of typical operations of target applications to engineer the value of $T_{backpressure}$.

With this probabilistic multipath forwarding, we can differentiate packets with different reliability requirements and also the probability that a packet reaches the destination is likely higher than its requirement.

3.3 Discussion of Concurrent Timeliness and Reliability

By combining aforementioned timeliness and reliability guarantee mechanisms, we expect that our proposed MMSPEED protocol can serve various packets with different timeliness and reliability requirements. Once a sensor node detects an event, it creates a packet x to be reported to the sink node. Based on the content of the sensor data, the source node selects the appropriate end-to-end deadline, $deadline(x)$, and required reaching probability, $P^{req}(x)$. The packet with end-to-end deadline and required reaching probability is forwarded towards its destination by MMSPEED. MMSPEED first classifies the packet into the proper speed layer based on the end-to-end deadline and the geographic distance to the destination as explained in Section 3.1. Then, the corresponding speed layer module l finds multiple forwarding nodes among those with progress speed higher than $SetSpeed_l$ such that the total reaching probability is higher than or equal to the required

reaching probability as explained in Section 3.2. Then, the packet is delivered to the chosen forwarding nodes.

When we deliver a packet to multiple nodes, it is important to ensure "parallel progress" along multiple paths so that each copy can meet the end-to-end deadline. Sending copies one by one to chosen neighbors may cause the later transmitted copy to miss the deadline even though the following nodes can guarantee the progress speed. For this reason, it is important to deliver a packet to multiple nodes using MAC layer multicast service based on broadcast nature of wireless medium rather than multiple calls of MAC unicast service.[4] We will discuss this MAC multicast in Section 4.

Since each copy denoted by x_c of a packet x progresses in parallel and its progress speed is guaranteed by the network-wide speed mechanism, the copy that eventually reaches the destination can meet the deadline with a high probability. This can be rephrased with a conditional probability—the probability that a copy x_c meets the deadline $deadline(x)$ under the condition of reaching the destination is approximately 1.0,

$$P(e2eDelay(x_c) \leq deadline(x) | x_c \text{ reaches the destination}) \approx 1.$$

From now on, for the simplicity of equations, we will use $x_c^{deadline}$ to represent the condition

$$"e2eDelay(x_c) \leq deadline(x)"$$

and x_c^{reach} for the condition "x_c reaches the destination." Also, the number of copies of a packet is determined in a way that the total reaching probability TRP is greater than or equal to the required reachability. Thus, the probability that at least one copy reaches the destination before the deadline can be derived as follows:

$$\begin{aligned}
&P(\text{at least one copy reaches destination before deadline}) \\
&= 1 - \prod_{\forall x_c} (1 - P(x_c^{deadline} \text{ AND } x_c^{reach})) \\
&= 1 - \prod_{\forall x_c} (1 - P(x_c^{deadline} | x_c^{reach}) P(x_c^{reach})), \\
&\text{since } P(x_c^{deadline} | x_c^{reach}) \approx 1, \\
&P(\text{at least one copy reaches destination before deadline}) \\
&\approx 1 - \prod_{\forall x_c} (1 - P(x_c^{reach})).
\end{aligned}$$

Note that $1 - \prod_{\forall x_c}(1 - P(x_c^{reach}))$ is TRP in (6) which is greater than or equal to the required reaching probability $P^{req}(x)$. Thus, we can meet the combined metric of *on-time reachability*—the probability that a packet reaches its final destination within deadline.

4 MAC Layer Features to Support MMSPEED Routing Protocol

Our proposed MMSPEED protocol alone cannot provide differentiated QoS guarantees. The proposed MMSPEED protocol relies on the premise that the underlying MAC protocol can perform the following functions:

- Prioritized access to shared medium depending on the speed layer,
- Reliable (or partially reliable) multicast delivery of packets to multiple neighbors,
- Supporting measurement of average delay to individual neighbors,
- Supporting measurement of loss rate to individual neighbors.

This section proposes extension of existing MAC protocols to best support MMSPEED routing protocol since none of current MAC protocols [19], [20], [21], [22] can fully support the above requirements. The goal of this section is to propose required add-on features for MMSPEED, instead of reinventing a totally new MAC protocol from the scratch. Thus, we mostly rely on EDCF (Enhanced Distributed Coordination Function) mode of IEEE 802.11e standard [22] and use the standard values if not mentioned specifically. The EDCF mode is most appropriate for ad hoc sensor networks since nodes can be deployed and work in an ad hoc way without any special access points.

As in IEEE 802.11e standard, the prioritization is achieved by differentiating interframe spacing (IFS) and backoff intervals for different classes. The basic idea is to use shorter IFS and backoff interval to higher priority class packets so that they can have higher chances to access the shared medium than lower priority class packets [23], [24], [19]. Each speed layer of MMSPEED is mapped to one MAC priority class, i.e., highest-speed to highest priority, second speed to second priority, and so on. This way, we can minimize internode priority inversion such that a high-speed packet in one node is not likely blocked by a low-speed packet in another node, realizing speed layer isolation as mentioned previously in Section 3.1.

Along with the prioritization, our MAC protocol maintains the average delay to each neighbor at each priority level. Specifically, in node i, when a request comes from MMSPEED to send a packet to neighbor j with priority-level l, a time stamp t_1 is associated with it. When node i receives ACK for the packet from node j, another time stamp t_2 is attached. Using t_1 and t_2, the MAC layer delay Δt can be calculated by:

$$\Delta t = t_2 - t_1 - SIFS - ACK, \tag{7}$$

where SIFS is the Short Interframe Spacing between the data and acknowledgment frames and ACK is the transmission delay of the acknowledgement frame. With this delay measurement, we maintain the exponential moving average of MAC layer delay to neighbor j at priority-level l. This MAC layer delay is included in the overall progress delay $delay_{i,j}^{l}$ that is used by MMSPEED in Section 3.1 for estimating the progress speed with speed-level l to select feasible forwarding nodes.

A more challenging problem is the reliable multicast support for multipath forwarding of MMSPEED. One simple approach is to repeatedly use the unicast sequence of RTS/CTS/DATA/ACK for reliable transmission to all recipients. However, it violates the "parallel progress" property by serializing the transmissions. Hence, later transmitted copies may experience longer delays, eventually missing their deadlines. The other extreme approach

4. A unicast based multipath forwarding also consumes more wireless channel resources.

is to simply use the broadcast nature of shared medium by transmitting a packet without RTS/CTS and ACK. If all the designated recipients can hear the packet successfully, all the copies received by the recipients can progress in parallel along multiple paths. However, without RTS/CTS and ACK, the probability of delivery success is very low.

Our MAC protocol aims to keep a balance between these two extremes. We select one of the recipients as the *primary recipient*, which will respond to the RTS frame with the CTS frame. Since the routing is performed based on the geographic information, we expect that there is high correlation among the locations of the multicast frame recipients and, thus, a single CTS frame provides a solution to the hidden node problem for most recipients with a high probability. Furthermore, only the primary recipient has the responsibility to acknowledge a received frame. Consequently, the sender node waits for ACK only from the primary recipient. If a timer expires before the acknowledgement, the sender retransmits up to MAX retransmission times before dropping the frame. Thus, in the timing perspective, it is like RTS/CTS/DATA/ACK unicast sequence except that the designated recipients eavesdrop the data. This multicast mode, which we call a *partially reliable multicast* mode, guarantees reliable transfer to the primary recipient only. Secondary recipients never obtain the possibility of having their frames retransmitted unless they eavesdrop the retransmissions for the primary recipient. However, we can expect that secondary recipients can receive the frame containing the packet with similar probabilities as the primary recipient due to their geographic correlation.

Even though a secondary recipient does not respond to RTS and DATA, it counts the number of received frames and reports the number to the sender whenever it is selected as the primary recipient. This report is used by the sender to estimate the MAC layer loss rate. The sender node also keeps track of the number of frames it sends to each of their neighbors as secondary recipients. When the sender receives the report piggybacked in ACK frame from a recipient, it updates the exponential moving average of the loss rates to the recipient either as primary or secondary. After these calculations, both counters at the sender and the primary recipient are reset to zero.

This MAC layer loss rate is included in the overall loss rates from node i to node j, $e_{i,j}^{primary}$ for primary recipient case and $e_{i,j}^{secondary}$ for secondary recipient case, which are used by MMSPEED in Section 3.2 for estimating the number of forwarding nodes to meet the required reaching probability. MMSPEED selects neighbor nodes as primary recipients in a round-robin manner so that each neighbor can report its status quite frequently without starvation.

5 Framework for Optimal Protocol Configuration

Until now, we explained our MMSPEED protocol assuming that the number of speed layers L and preset speed value $SetSpeed_l$ for each speed layer l $(1 \leq l \leq L)$ are given a priori. This section provides a guideline to determine L and $SetSpeed_l$ for optimal configuration of MMSPEED in the offline design phase.

The more speed layers we have, the finer service differentiation is possible. However, this incurs a larger protocol overhead in both processing power and memory requirements. Thus, the maximum number of speed layers, i.e., L, can be determined considering the practical limits of processing power and memory size of sensor nodes.

Once L is determined, the remaining design problem is to determine the preset speed values

$$SetSpeed_1, SetSpeed_2, \cdots, SetSpeed_L.$$

This problem turns out to be a trade-off problem of two correlated quality domains, timeliness and reliability. Let us first give only a high-level intuition on this trade-off issue. In order to improve the reliability, we may want to drop less packets and retransmit packets at each hop if error occurs. However, this increases the overall workload and hence degrades the timeliness quality, resulting in longer delay. On the other hand, dropping more packets sacrificing the reliability will result in less workload[5] and, thus, shorter delay, improving the timeliness quality. Since MMSPEED resembles the SPEED mechanisms for each speed layer, it probabilistically drops packets in the network layer if nodes cannot forward packets with progress speeds higher than preset speed $SetSpeed_l$. Thus, the preset speed $SetSpeed_l$ affects both timeliness and reliability. Intuitively, if we set $SetSpeed_l$ high, we need to drop more packets to maintain low workload to ensure the high preset speed network-widely for layer l,

$$SetSpeed_l \propto PacketDropProb_l \propto \frac{1}{reliability_l}.$$

This shows the inverse-proportional relation between timeliness and reliability. Therefore, we need to set the speed values by optimally trading off these two inversely proportional QoS domains.

This trade-off can be demonstrated with a simple example with two speed layers, i.e., $L = 2$. Predicting the typical usage of a sensor network, the sensor network designer can estimate the workload characteristics such as probabilistic distribution of event locations and required end-to-end deadlines. For example, Fig. 6 may be a spatial distribution of events for monitoring wildlife along the middle line of the forest. Fig. 7 may be another example of spatial distribution of events for intruder tracking mostly at a border.

From the event distribution models, we can derive the probability function of required speed $Prob(ReqSpeed)$ as follows: 1) From the location of an event (i.e., the source of a packet), we can estimate the distance to the final destination $dist_{s,d}$ and divide it by the event-associated deadline D to calculate the required speed as $ReqSpeed = \frac{dist_{s,d}}{D}$. 2) From the spatial distribution of each event type (e.g., Fig. 6 and Fig. 7) and the above calculated $ReqSpeed$ for each event instance (i.e., small circles in Fig. 6 and Fig. 7), we can derive the probability function of $ReqSpeed$ for each event type (e.g., Fig. 8a and Fig. 8b for event spatial distributions

5. We do not consider retransmission from the sources to recover lost packets since it is not appropriate in real-time sensor networks.

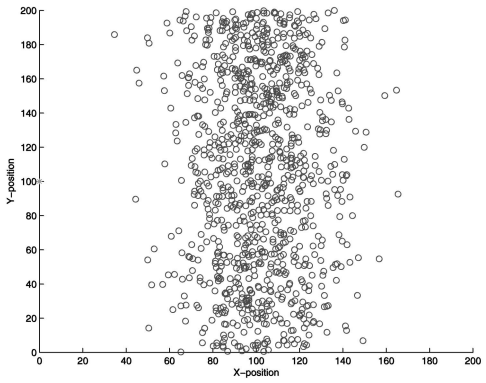


Fig. 6. Event Type 1 spatial distribution.

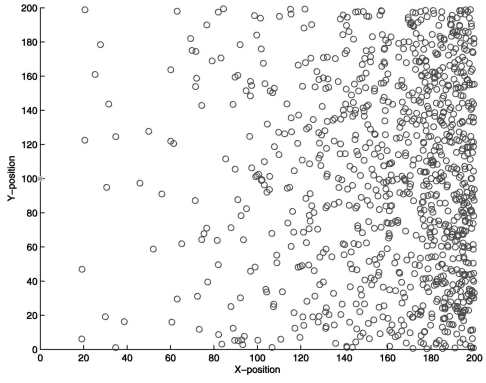


Fig. 7. Event Type 2 spatial distribution.

of Fig. 6 and Fig. 7, respectively). The probability function gives the probability that a packet triggered by an event of each type will require a certain value of $ReqSpeed$ to meet its associated deadline. 3) Finally, we combine probability functions of all event types and derive the combined probability function $Prob(ReqSpeed)$ as in Fig. 8c.

From now on, let us consider Fig. 9 as such calculated probability function $Prob(ReqSpeed)$. Note that the absolute amount of workload $W(ReqSpeed)$ that requires $ReqSpeed$ to meet the deadline can be represented by $Prob(ReqSpeed)$ multiplied by the workload scale factor f:[6]

$$W(ReqSpeed) = f \cdot Prob(ReqSpeed).$$

Our goal is to find $SetSpeed_1$ and $SetSpeed_2$ (when $L = 2$) that maximizes the affordable workload (i.e., maximize the scale factor f) while meeting both timeliness and reliability requirements. The maximum required speed, $maxReqSpeed$ is determined by the longest end-to-end distance—size of sensor network terrain and the shortest end-to-end deadline, which are given in the offline design phase. To support such requests, the highest preset speed $SetSpeed_1$ needs to be configured as $maxReqSpeed$. Now, $SetSpeed_2$ is a control nob ranging from $minReqSpeed$ to $maxReqSpeed$. If we move $SetSpeed_2$ right (raise the delivery speed of Layer 2), a large amount of workload belongs to Layer 2 and, thus, a large number of packets need to be dropped for network-wide guarantee of the high value of $SetSpeed_2$. This sacrifices the reliability of packets belonging to Layer 2. On the other hand, if we move $SetSpeed_2$ to left (lower the delivery speed of Layer 2), the amount of workload belonging to Layer 2 decreases and thus the low value of $SetSpeed_2$ can easily be network-widely guaranteed without dropping many packets. However, this makes a large portion of workload belong to Layer 1 and, thus, many packets of Layer 1 need to be dropped to network-widely guarantee $SetSpeed_1$. In general, decreasing $SetSpeed_i$ (move to left) improves the reliability of Layer i but sacrifices the reliability of Layer $i-1$. Thus, we have to find the optimal trade-off between timeliness and reliability not only within a single layer but also across multiple layers.

6. To represent both $Prob(ReqSpeed)$ and $W(ReqSpeed)$ in a single graph, the y-axis of the figure has double meanings—left-y-axis is probability density of required speed $Prob(ReqSpeed)$ and right-y-axis is the absolute workload amount that require $ReqSpeed$, i.e., $W(ReqSpeed)$.

Our idea to find such trade-off is a *model-based trade-off* that uses an analytical model of workload versus delay and balances the workload of each layer by controlling $SetSpeed$ values. Let us explain this again with the two layer example. Using the existing analytical performance models for the MAC 802.11 protocol [25], [26], [27], [28], we can calculate the average one-hop delay $Delay_l(W_l)$ as a function of the workload W_l that belongs to Layer l. Fig. 10a and Fig. 10b show such analytic models for Layer 1 and Layer 2. From the average one-hop delay, we can estimate the average packet delivery speed we can get $GetSpeed_l$ by dividing the average one-hop progress distance by the average one-hop delay,

$$GetSpeed_l(W_l) = \frac{OneHopProgressDistance}{Delay_l(W_l)}, \qquad (8)$$

where $OneHopProgressDistance$ can be approximately estimated by the average node density [29]. Thus, the workload versus delay models in Fig. 10a and Fig. 10b can be converted to the workload versus $GetSpeed$ curves as in Fig. 10c and Fig. 10d. Note that $GetSpeed$ for each layer depends on the amount of workload belonging to the corresponding layer. Also, the amount of workload W_1 and W_2 belonging to Layer 1 and Layer 2, respectively, are functions of $SetSpeed_2$ as shown in Fig. 9:

$$W_1(SetSpeed_2) \propto \frac{1}{SetSpeed_2} \text{ and}$$
$$W_2(SetSpeed_2) \propto SetSpeed_2.$$

If we use a high value of $SetSpeed_2^H$, i.e., increase the preset speed for the Layer 2, the share of the workload $W_1(SetSpeed_2^H)$ for Layer 1 becomes small as we can see in Fig. 9. A reduced workload would result in lower delay in the Layer 1 and the resulting $GetSpeed_1$ can be much higher than the targeted $SetSpeed_1$ as shown by the X mark in Fig. 10c. This means that maintaining the $GetSpeed_1$ above $SetSpeed_1$ will require fewer packet drops, resulting in high reliability of Layer 1. Thus, the gap $(GetSpeed_1 - SetSpeed_1)$ can be interpreted as the *reliability margin of Layer 1*—the larger the margin, the smaller is the probability of packet drops to meet $SetSpeed_1$. However, the high value $SetSpeed_2^H$ results in a large workload $W2(SetSpeed_2^H)$ for Layer 2 as we can see in Fig. 9. Consequently, $GetSpeed_2$ is much lower than $SetSpeed_2^H$ as shown by the X mark in Fig. 10d—negative

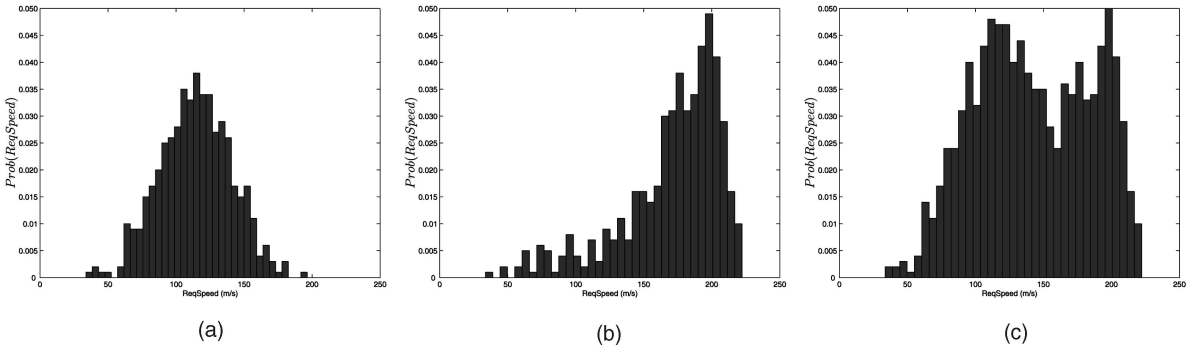


Fig. 8. Probability function of $ReqSpeed$. (a) $Prob(ReqSpeed)$ for Event Type 1. (b) $Prob(ReqSpeed)$ for Event Type 2. (c) Combined $Prob(ReqSpeed)$.

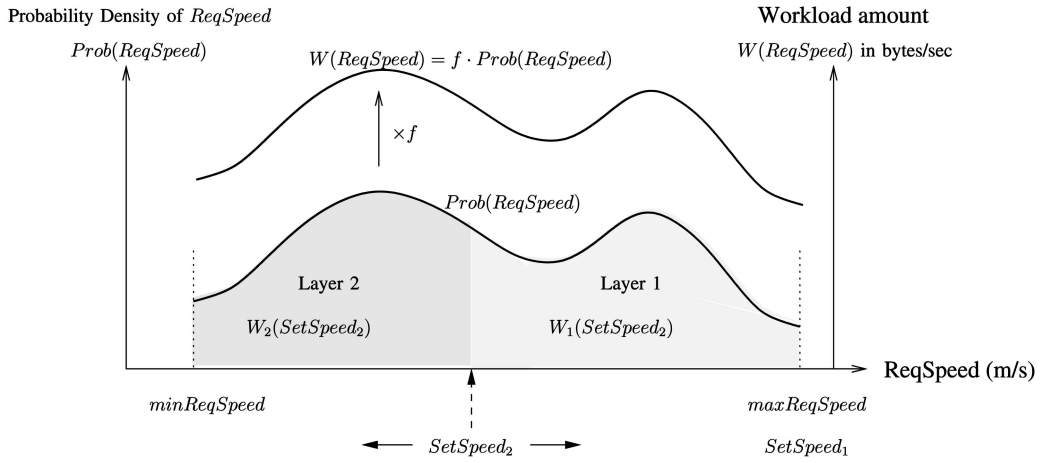


Fig. 9. Probability density function and workload amount of required speed.

reliability margin for Layer 2. This implies that many packets of Layer 2 will be dropped to maintain $SetSpeed_2^H$. This solution is biased in the sense that the Layer 1 is given unnecessary incentives at the expense of the Layer 2 reliability.

To have a balanced solution, we can gradually reduce $SetSpeed_2$ until the workload can be partitioned appropriately such that the reliability margins for Layer 1 and Layer 2 are as balanced as possible like $SetSpeed_2^L$ denoted by the O marks in Fig. 10c and Fig. 10d. This way we can accommodate the largest amount of workload (i.e., the largest workload scale factor f) meeting all preset speeds $SetSpeed_1$ and $SetSpeed_2$ while maintaining a small drop probability for Layer 1 and Layer 2.

Generalizing this intuition, our problem can be formally described as follows:

Problem. Find the preset speeds $SetSpeed_l$ $(1 \leq l \leq L)$ such that the workload scale factor f can be maximized under the constraint of positive re-liability margin $GetSpeed_l - SetSpeed_l$ for all speed layers l $(1 \leq l \leq L)$:

$$\begin{aligned} &\text{Maximize} \quad f \\ &\text{Subject to} \quad GetSpeed_l - SetSpeed_l > 0, \text{for all } l \in \{1, \cdots, L\}. \end{aligned}$$

To solve this optimization problem, we propose a heuristic approach that incrementally finds the preset speeds from $SetSpeed_1$ to $SetSpeed_L$ and performs an outer loop to maximize f. Let us first explain how to find $SetSpeed_l$ $(1 \leq l \leq L)$ incrementally ensuring positive reliability margins $GetSpeed_l - SetSpeed_l$ when the workload scale factor f is given. The following pseudocodes describes such algorithm called **FindSetSpeed(f)**:

FindSetSpeed(f): For a given f, find $SetSpeed_l$ $(1 \leq l \leq L)$ such that $GetSpeed_l > SetSpeed_l$ for all l
Input: f
Output: $result$—$success$ if solutions found, $fail$ otherwise
$SetSpeed_1, SetSpeed_2, \cdots, SetSpeed_L$

```
begin procedure
1.  SetSpeed_1 = maxReqSpeed
2.  for l = 2 to L do
3.    SetSpeed_l = SetSpeed_{l-1}
4.    while(1) do
5.      W_{l-1} = f · Prob(SetSpeed_l ≤ ReqSpeed ≤
        SetSpeed_{l-1}) /* see Figure 9 */
6.      GetSpeed_{l-1} = calcGetSpeed(W_{l-1}) /* see Figure 10*/
7.      if (GetSpeed_{l-1} > SetSpeed_{l-1})
8.        SetSpeed_l = SetSpeed_l − δ
9.      else
10.       SetSpeed_l = SetSpeed_l + δ /* restore previous
          value*/
11.       break
12.     end if
13.   end while
14. end for
```

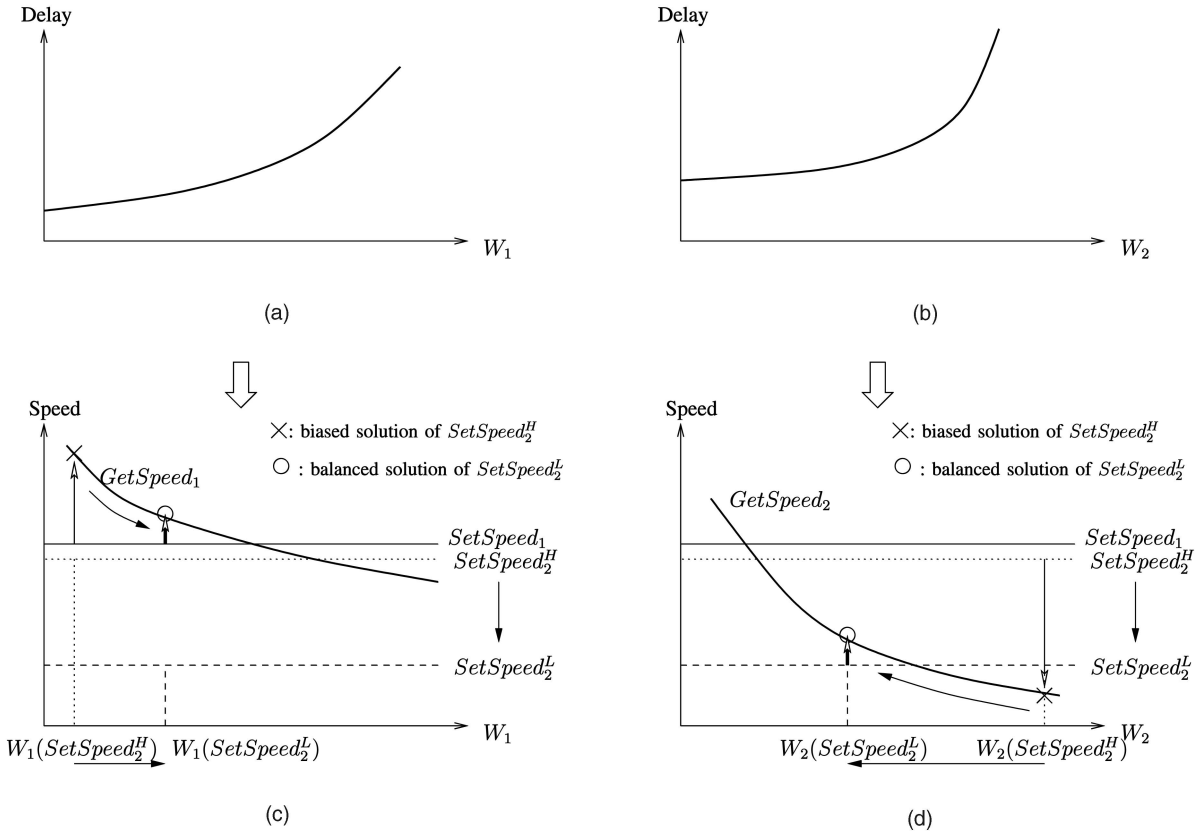


Fig. 10. Analytical model for workload partition. (a) Workload versus delay for Layer 1. (b) Workload versus delay for Layer 2. (c) Workload versus $GetSpeed$ for Layer 1. (d) Workload versus $GetSpeed$ for Layer 2.

15. $W_L = f \cdot Prob(minReqSpeed \leq ReqSpeed \leq SetSpeed_L)$
16. $GetSpeed_L = calcGetSpeed(W_L)$
17. **if** $(GetSpeed_L > SetSpeed_L)$
18. $\quad result = success$
19. **else**
20. $\quad result = fail$
21. **end if**
22. **return** $result, (SetSpeed_1, \cdots, SetSpeed_L)$

end procedure

This procedure first initializes $SetSpeed_1$ as $maxReqSpeed$ in line 1. Then, the for loop from line 2 to line 14 incrementally finds $SetSpeed_2, \cdots, SetSpeed_L$. To find $SetSpeed_l$ after fixing $SetSpeed_{l-1}$, line 3 first sets $SetSpeed_l$ as $SetSpeed_{l-1}$ and then the while loop from line 4 to line 13 gradually decreases $SetSpeed_l$. Decreasing $SetSpeed_l$ will increase the workload W_{l-1} as explained in Fig. 9. The workload W_{l-1} can be calculated as in line 5 from the given workload model of Fig. 9. For W_{l-1}, line 6 can calculate $GetSpeed_{l-1}$ from the workload versus $GetSpeed$ model in Fig. 10. If $GetSpeed_{l-1} > SetSpeed_{l-1}$, we keep decreasing $SetSpeed_l$ in lines 7 and 8 to further increase Layer $l-1$ workload W_{l-1}. If $GetSpeed_{l-1}$ becomes less than or equal to $SetSpeed_{l-1}$, we terminate the while loop in lines 9, 10, and 11, after restoring the last value of $SetSpeed_l$, which makes the largest workload of Layer $l-1$ ensuring the positive reliability margin of Layer $l-1$. At line 15, we found preset speed values up to $SetSpeed_L$ such that the positive reliability margin can be ensured up to Layer $L-1$. Now, all the remaining workload belongs to Layer L as in line 15. With the workload W_L, we can calculate $GetSpeed_L$ in line 16. If $GetSpeed_L > SetSpeed_L$, the solution is feasible ensuring positive reliability margin for all layers and, thus, we return $success$ in line 18. Otherwise, the workload scale factor f is too large to ensure the positive reliability margins for all layers and, thus, we return $fail$ in line 20.

By repeatedly using **FindSetSpeed(f)**, we can find the maximum workload scale factor f and its corresponding preset speed values. This outer loop procedure is described as follows:

MaximizeF: Find the maximum f and the corresponding preset speed values $SetSpeed_l$ $(1 \leq l \leq L)$

begin procedure

1. $f = 0$
2. **repeat**
3. $\quad f = f + \delta$
4. $\quad \{result, SetSpeed_l, \cdots, SetSpeed_L\} = FindSetSpeed(f)$
5. **until** $result == fail$
6. $f = f - \delta$
7. $\{result, SetSpeed_l, \cdots, SetSpeed_L\} = FindSetSpeed(f)$
8. **return** $(SetSpeed_1, \cdots, SetSpeed_L)$

end procedure

This procedure starts with $f = 0$ in line 1. Then, the loop from line 2 and line 5 repeatedly calls **FindSetSpeed(f)** while increasing f by δ at each iteration. When **FindSetSpeed(f)** returns the first $fail$ signal, $f - \delta$ is the maximum

TABLE 1
Simulation Environment Settings

Bandwidth	200 Kbps
Payload	32 bytes
Terrain	200m×200m
Node number	100
Node Placement	Uniform
Radio Range	40 m

TABLE 2
EDCF MAC Parameters

Power Saving Mode	Disabled
Fragmentation	Disabled
Retry Limit (MAX)	7
Priority Classes	2
SIFS	$10\mu s$
Time Slot	$20\mu s$
AIFS[1], AIFS[2]	2, 4 time slots
CW_{min}[1], CW_{min}[2]	15, 31 time slots
CW_{max}[1], CW_{max}[2]	255, 511 time slots
Persistent Factor	2

workload scale factor for which feasible solution of $SetSpeed_l$ $(1 \leq l \leq L)$ exists. The procedure returns such found preset speeds in line 8.

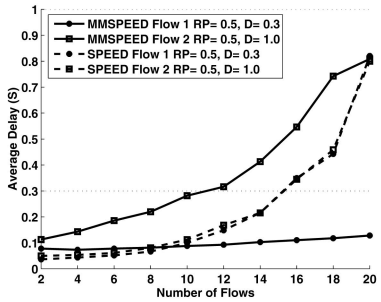
These two procedures **FindSetSpeed(f)** and **MaximizeF** provide a sound design guideline for MMSPEED configuration by optimally trading-off reliability and timeliness across multiple speed layers.

6 Experimental Results

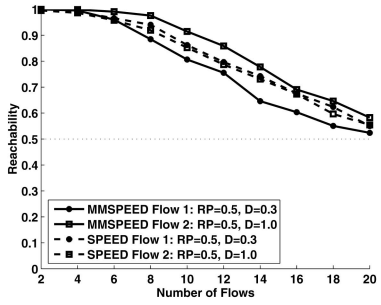
We conducted extensive simulations of the proposed MMSPEED protocol using J-SIM network simulator [30] and its performance is compared with SPEED [9], which is the only protocol available in the literature that can provide real-time services in a localized way in sensor networks. The general simulation environment is mainly drawn from [9] for fair comparison and summarized in Table 1. Table 2 shows the 802.11e EDCF MAC parameters used in the simulation.

6.1 Service Differentiation and Guarantee

In the first experiment, we show the service differentiation in the timeliness domain by MMSPEED. In order to focus on the timeliness domain, we use the same and nonstrict reliability requirement of 0.5 for all n flows. However, we divide n flows into two groups: one (flow group 1) has a strict real-time requirement, i.e., short end-to-end deadline of 0.3 sec and the other (flow group 2) has long end-to-end deadline of 1.0 sec. In MMSPEED protocol, we use two



(a)



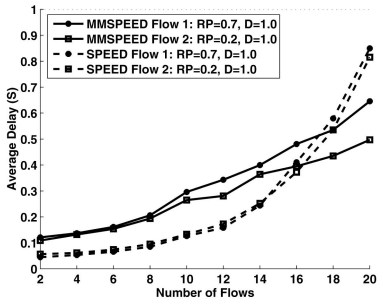
(b)

Fig. 11. Timeliness differentiation. (a) Average delay. (b) Reaching probability.

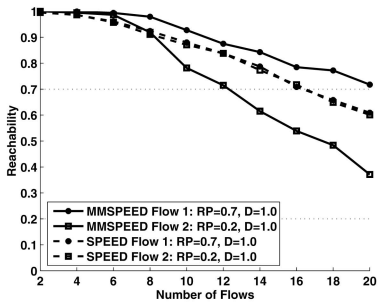
speed levels of 1,000 m/sec and 250 m/sec while the SPEED protocol uses one highest speed level 1,000 m/sec to meet the most urgent packet requirement. Fig. 11a shows the average end-to-end delay for each flow group as increasing the number of flows n (solid lines for MMSPEED and dashed lines for SPEED). The figure shows that MMSPEED can provide clear differentiation of delay for two groups of flows with different end-to-end deadline requirements. As a result, the average end-to-end delay for each group is under the end-to-end deadline up to 20 flows. On the other hand, SPEED protocol cannot differentiate the two flow groups and thus the average delay for flow group 1 is under the deadline 0.3 sec only up to 14 flows.

Fig. 11b shows the reachability for each group of flows by each protocol. There is no big performance difference in the reliability domain since every flow has the same reliability requirement in this experiment.

In the second experiment, we show the capability of our protocol to differentiate services in the reliability domain. For this, we use two flow groups with different reliability requirements (flow group 1—high reliability of 0.7 and flow group 2—low reliability of 0.2) but the same deadline requirement of 1 sec. As before, MMSPEED has two preset speed levels of 1,000 m/sec and 250 m/sec. For SPEED protocol, we used low speed level 250 m/sec so that less packets need to be dropped for speed guarantee, which gives the favor to SPEED in the reliability domain. Fig. 12b shows that MMSPEED can provide clear service differentiation in the reliability domain and, thus, both flow groups can meet their own reliability requirements up to 20 flows. On the other hand, in SPEED protocol, two flow groups are

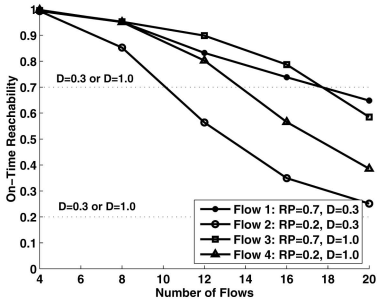


(a)

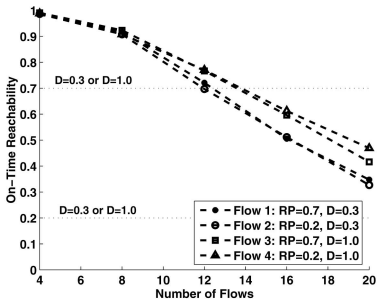


(b)

Fig. 12. Reliability differentiation. (a) Average delay. (b) Reaching probability.



(a)



(b)

Fig. 13. On-time reachability. (a) MMSPEED. (b) SPEED.

mixed up with no differentiation, which makes flow group 1 miss reliability requirement of 0.7 for 18 flows and more.

Fig. 12a shows the average delay as a reference. No big difference in delay can be observed since all flows have the same deadline requirement. The average delay for each flow group in each protocol is much lower than the nonstrict end-to-end deadline requirement of 1.0 sec.

In the previous two experiments, we showed MMSPEED's capability for service differentiation in timeliness and reliability domain. The service differentiation, however, does not imply the end-to-end service guarantee in the combined metric of on-time reachability. To justify the MMSPEED protocol in the sense of guaranteeing the end-to-end on-time reachability, we conduct another experiment with mixed traffics. For this, we divide the n flows into four groups:

1. flow group 1 with short deadline 0.3 sec and high reachability 0.7,
2. flow group 2 with short deadline 0.3 sec and low reachability 0.2,
3. flow group 3 with long deadline 1.0 sec and high reachability 0.7, and
4. flow group 4 with long deadline 1.0 sec and low reachability 0.2.

Fig. 13a and Fig. 13b show the on-time reachability for each flow group by MMSPEED and SPEED, respectively. In MMSPEED, the flow groups 1 and 3 with high on-time reachability requirements can actually achieve high on-time reachability meeting the requirements up to 18 flows. The flow groups 2 and 4 with lower on-time reachability requirement can get less favor but still meet the requirements up to 20 flows. Summarizing Fig. 13a, MMSPEED can afford up to 18 flows meeting on-time reachability of all flows. On the other hand, SPEED protocol can afford only up to 12 flows meeting on-time reachability of all flows as shown in Fig. 13b. This is because SPEED protocol mixes up all the flows without any differentiation depending on timeliness and reliability requirements.

6.2 Workload Overhead Analysis

This section compares the overhead of MMSPEED and SPEED protocols. We consider two types of overhead. The first type is the control overhead that includes 1) location update packets periodically broadcast to immediate neighbors, 2) timeliness back-pressure packets for speed control, and 3) reliability back-pressure packets for reliability control. The first two control packets are required by both MMSPEED and SPEED protocols while the third control packets are required only by MMSPEED. The second type is data packet multiplication overhead required for leveraging multipath routing by MMSPEED.

Fig. 14a and Fig. 14b show the overhead of each protocol as increasing the number of flows. The flows are divided into four groups with different deadline and reliability requirements as in Fig. 13. Fig. 14a shows the total numbers of control packets generated by MMSPEED and SPEED for the whole duration of simulation. Unlike our simple intuition, the total number of control packets by MMSPEED is lower than SPEED. This can be explained as follows: 1) the number of periodic location update packets is same for MMSPEED and SPEED, 2) the number of timeliness back-pressure packets in SPEED is larger than MMSPEED since only half of traffic needs to use high speed class in MMSPEED while all traffic competes for the high speed

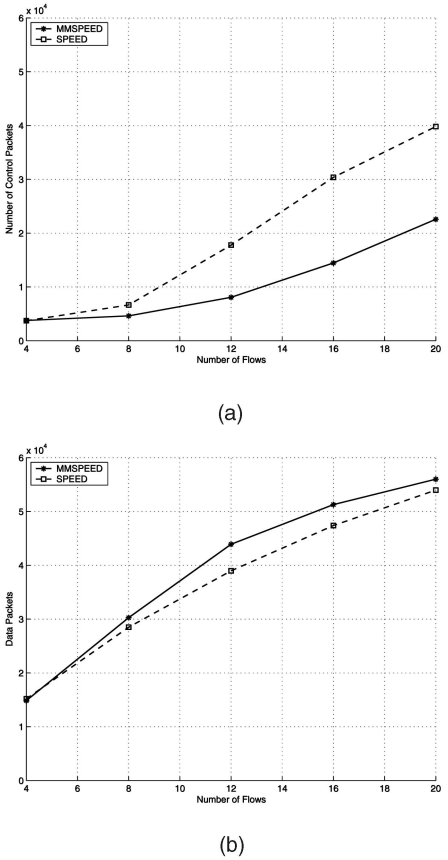


Fig. 14. Overhead versus number of flows. (a) Control packets. (b) Data packets.

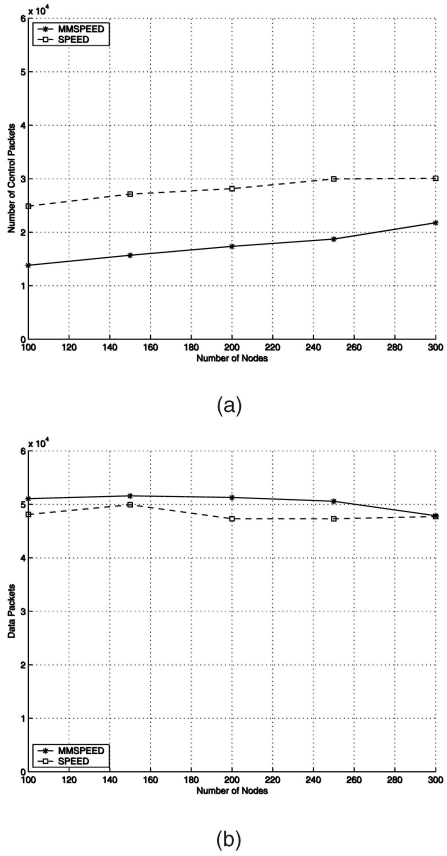


Fig. 15. Overhead versus node density. (a) Control packets. (b) Data packets.

class in SPEED resulting in many back-pressure packets, and 3) the number of reliability back-pressure packets which is an extra overhead of MMSPEED is quite small compared to the number of timeliness back-pressure packets.

To show the data packet multiplication overhead, Fig. 14b shows the total numbers of data packets transmitted all over the network during the duration of simulation. The additional data packet transmissions for multipath routing in MMSPEED is surprisingly small compared to the total packet transmission in SPEED. This can be explained as follows: First, MMSPEED makes a just adequate number of packet multiplications mostly at early stages of route paths preventing exponential packet multiplications. Second, many copies of packets are dropped at early stages of route paths since those copies are assigned with smaller reliability requirements than the original one. This can bound the total number of packet transmissions. Finally, MMSPEED differentiates packets depending on their reliability requirement and, thus, we can drop more packets with low reliability requirements than SPEED, giving more resource for multipath forwarding of packets with high reliability requirements. Therefore, the aggregated number of packet transmissions of MMSPEED becomes comparable with SPEED.

In order to see the scalability of MMSPEED, we measure the overhead as increasing the node density. Starting from the total 100 nodes, we incrementally add nodes at random locations. Fig. 15a shows that the total number of control packets only linearly increases in both MMSPEED and SPEED mainly due to the location update packets periodically generated at each node. Such linear increase of control packets is the advantage of localized routing protocols, which makes them scalable. On the other hand, proactive or reactive routing protocols utilizing global topology information cause an exponential increase of control packets as increasing the node density. As we can see in Fig. 15b, the number of data packet transmissions is generally constant in both MMSPEED and SPEED because both protocols can manage the similar hop counts regardless of node density using the geographic forwarding node selection. This is another nice property of MMSPEED and SPEED for the scalability.

6.3 Adaptability to Dynamic Topology Changes

Until now, we use static network topology where each node is placed at a fixed position. In order to show the adaptability of MMSPEED to the dynamic topology changes, we conduct another experiment. In this experiment, we use a network with 150 nodes randomly placed and 12 flows divided into four groups with different requirements as before. All other parameters are same as the previous experiments. For the initial 400 sec, the network is static. At the time instant of 400 sec, 20 percent of nodes start moving randomly. Those nodes are in motion for the next 200 sec. After that, they stop moving. For the

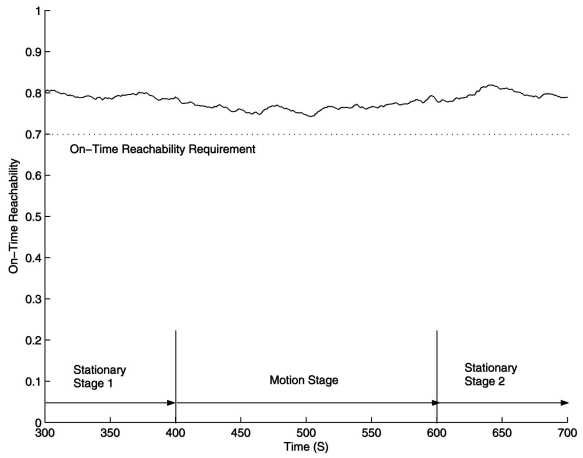


Fig. 16. Adaptability of MMSPEED.

whole duration of simulation, we measure the on-time reachability with moving window of 1,000 packets.

Fig. 16 shows the time trace of on-time reachability for only one flow with most strict requirements, i.e., deadline of 0.3 sec and reliability of 0.7 among all four flow groups. From this graph, we observe that MMSPEED can guarantee the on-time reachability not only for the stationary stage but also for the motion stage continuously adapting to network topology changes. The resulting on-time reachability in the motion stage is a little bit lower than that in the stationary stage. This on-time reachability loss in the motion state is because of the gap between node's neighbor table and the actual locations of neighbors since a node can notice that an existing neighbor leaves its radio range only after timeout without receiving location update packets from the neighbor. During the period of such gap, a node can forward packets to a node that is not within the radio range resulting in packet losses. We can reduce the period of such misforwarding by increasing the location update frequency. However, this in turn increases the control overhead. Thus, it is a design issue to select a proper location update frequency by trading-off the adaptability and control overhead.

6.4 Discussion on Power Overhead

The nice performance of MMSPEED observed in the previous sections cannot be achieved for free. Many features of MMSPEED may lead to more energy consumption due to more complex computation and longer frame with overhead bits. Although the power consumption problem is not the main focus of this paper, this section presents the power-related overhead of MMSPEED to see the potential of applying MMSPEED to power-constrained scenarios.

Basically, there are two main sources of power consumption in sensor networks: 1) sensor node processors and 2) sensor node radio modules. Since MMSPEED requires more complex calculations for multiple path selection and dynamic compensation, it may lead to more power consumption by processors. However, according to previous studies [31], [32], [33], the dominant source of power consumption is the radio module. Thus, we can expect that the increased computational complexity of MMSPEED does not severely increase the total power consumption.

TABLE 3
Multicast Overhead to RTS/CTS/DATA/ACK Frames Sizes (in Bytes)

Frame	802.11	MMSPEED
RTS	20	26
CTS	14	14
DATA	34	34
ACK	14	16

The real source of additional power overhead comes from the radio module due to the increased amount of data transmission by MMSPEED. First, MMSPEED uses multipath forwarding by transmitting duplicated copies of the same packets and also uses a larger number of hops rather than only shortest path. This would increase the overall power consumption by sensor node radio modules. However, MMSPEED makes use of the just adequate number of paths and hops necessary to meet the timeliness and reliability requirements, without severely overutilizing resources as discussed in Section 6.2. As a consequence, the total number of data transmissions by MMSPEED including duplicated copies over all hops is only slightly larger than that of SPEED as we can observe in Fig. 14b. This implies that the multipath forwarding of MMSPEED incurs only small increase of overall power consumption, which is worth to be paid for QoS provisioning. Second, to utilize the mutlipath forwarding, MMSPEED has multicast capability in the MAC layer. This features adds additional overhead bits in the RTS and ACK frames to exchange the additional information as described in Section 4. This may cause increased power consumption by radio modules but the overhead is small compared to the standard frame size as can be seen in Table 3.

7 Conclusion

In this paper, we propose a novel packet delivery mechanism called MMSPEED for wireless sensor networks to provide service differentiation and probabilistic QoS guarantees in the timeliness and reliability domains. For the timeliness domain, we provide multiple network-wide speed options so that various traffic types can dynamically choose the proper speed options for their packets depending on their end-to-end deadlines. For the reliability domain, we use probabilistic multipath forwarding to control the number of packet delivery paths depending on the required end-to-end reaching probability. These methods are implemented in a localized way with dynamic compensation to compensate for the inaccuracies of local decisions as packets progress towards their destinations. Since the proposed mechanisms work locally at each node without global network state information and end-to-end path setup, it can preserve desirable properties such as scalability for large sensor networks, self adaptability to network dy-

namics, and appropriateness for urgent aperiodic and periodic packets. Simulation results show that MMSPEED can efficiently cater for the needs of various traffic types with different combinations of reliability and timeliness requirements. As a result, MMSPEED can significantly improve the effective capacity of a sensor network in terms of number of flows meeting both reliability and timeliness requirements.

In the future, we will extend our work to the power consumption domain by conducting actual measurement of power consumption in the real-network and its detail analysis to revise MMSPEED for power-constrained applications.

ACKNOWLEDGMENTS

This work is supported in part by US National Science Foundation grant 0509175.

REFERENCES

[1] J. Zhao and R. Govindan, "Understanding Packet Delivery Performance in Dense Wireless Sensor Networks," *Proc. First Int'l Conf. Embedded Networked Sensor Systems,* Nov. 2003.

[2] T. Chen, J. Tsai, and M. Gerla, "QoS Routing Performance in Multihop Multimedia Wireless Networks," *Proc. IEEE Sixth Int'l Conf. Universal Personal Comm.,* vol. 2, pp 557-561, 1997.

[3] R. Sivakumar, P. Sinha, and V. Bharghavan, "CEDAR: Core Extraction Distributed Ad Hoc Routing Algorithm," *IEEE J. Selected Areas in Comm.,* vol. 17, no. 8, pp. 1454-1465, 1999.

[4] S. Chen and K. Nahrstedt, "Distributed Quality-of-Service Routing in Ad Hoc Networks," *IEEE J. Selected Areas in Comm.,* vol. 17, no. 8, pp. 1488-1505, 1999.

[5] B. Hughes and V. Cahill, "Achieving Real-Time Guarantees in Mobile Ad Hoc Wireless Networks," *Proc. Work-in-Progress Session 24th IEEE Real-Time Systems Symp.,* Dec. 2003.

[6] 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," *Proc. IEEE Real-Time and Embedded Technology and Applications Symp. (RTAS 2002),* Sept. 2002.

[7] S. Bhatnagar, B. Deb, and B. Nath, "Service Differentiation in Sensor Networks," *Proc. Fourth Int'l Symp. Wireless Personal Multimedia Comm.,* Sept. 2001.

[8] M. Caccamo, L. Zhang, L. Sha, and G. Buttazzo, "An Implicit Prioritized Access Protocol for Wireless Sensor Networks," *Proc. IEEE Real-Time Systems Symp. (RTSS '02),* pp. 39-48, 2002.

[9] T. He, J. Stankovic, C. Lu, and T. Abdelzaher, "SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks," *Proc. IEEE Int'l Conf. Distributed Computing Systems,* pp. 46-55, 2003.

[10] B. Deb, S. Bhatnagar, and B. Nath, "ReInForm: Reliable Information Forwarding Using Multiple Paths in Sensor Networks," *Proc. IEEE Int'l Conf. Local Computer Networks,* pp. 406-415, 2003.

[11] Q. Huang, C. Lu, and G.-C. Roman, "Spatiotemporal Multicast in Sensor Netwroks," *Proc. First Int'l Conf. Embedded Networked Sensor Systems,* Nov. 2003.

[12] B. Karp and H. Kung, "Greedy Perimeter Stateless Routing for Wireless Networks," *Proc. IEEE/ACM Int'l Conf. Mobile Computing and Networking,* pp. 243-254, 2000.

[13] T. He, C. Huang, B. Blum, J. Stankovic, and T. Abdelzaher, "Range-Free Localization Schemes for Large Scale Sensor Networks," *Proc. Mobicom Conf. ,* 2003.

[14] L. Doherty, K.S. J. Pister, and L.E. Ghaoui, "Convex Positiion Estimation in Wireless Sensor Networks," *Proc. Infocom Conf.,* 2001.

[15] Y.B. Ko and N. Vaidya, "Location-Aided Routing (LAR) in Mobile Ad Hoc Networks," *Proc. ACM/IEEE Fourth Int'l Conf. Mobile Computing and Networking,* Oct. 1998.

[16] I. Stojmenovic and X. Lin, "GEDIR: Loop-Free Location Based Routing in Wireless Networks," *Proc. IASTED Int'l Conf. Parallel and Distributed Computing and Systems,* Nov. 1999.

[17] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, "Highly-Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks," *ACM SIGMOBILE Mobile Computing and Comm. Rev.,* vol. 5, no. 4, pp. 11-25, 2001.

[18] S. Dulman, T. Nieberg, J. Wu, and P. Havinga, "Trade-Off between Traffic Overhead and Reliability in Multipath Routing for Wireless Sensor Networks," *Proc. Wireless Comm. and Networking Conf.,* 2003.

[19] R. Baldwin, N. Davis, S. Midkiff, and R. Raines, "Packetized Voice Transmission Using RT-MAC, a Wireless Real-Time Medium Access Control Protocol," *ACM Mobile Computing and Comm. Rev.,* vol. 5, no. 3, pp. 11-25, 2001.

[20] K. Tang and M. Gerla, "MAC Reliable Broadcast in Ad Hoc Networks," *Proc. IEEE Military Comm. Conf. (MILCOM 2001),* vol. 2, pp. 1008-1013, 2001.

[21] M. Sun, L. Huang, A. Arora, and T. Lai, "Reliable MAC Layer MMulticast in IEE 802.11 Wireless Networks," *Proc. IEEE Int'l Conf. Parallel Processing,* vol. 2, pp. 527-536, 2002.

[22] I. WG, "Draft Supplement to Standard for Telecommunications and Information Exchange between Systems—LAN/MAN Specific Requirements—Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Medium Access Control (MAC) Enhancements for Quality of Service (QoS)," IEEE 802.11e/D3.0, May 2002.

[23] W. Pattara-Atikom, P. Krishnamurthy, and S. Banerjee, "Distributed Meachanisms for Quality of Service in Wireless LANs," *IEEE Wireless Comm.,* vol. 10, no. 3, pp. 26-34 June 2003.

[24] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi, and E. Knightly, "Distributed Priority Scheduling and Medium Access in Ad Hoc Networks," *ACM/Kluwer Wireless Networks,* no. 8, pp. 466-466, 2002.

[25] G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function," *IEEE J. Selected Areas in Comm.,* pp. 535-547, Mar. 2000.

[26] O. Tickoo and B. Sikdar, "Queueing Analysis and Delay Mitigation in IEEE 802.11 Random Access MAC Based Wireless Networks," *Proc. IEEE INFOCOM Conf.,* Mar. 2004.

[27] M. Mzdemir and A.B. McDonald, "An m/mmgi/1/k Queuing Model for IEEE 802. 11 Ad Hoc Networks," *Proc. First ACM Int'l Workshop Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks,* pp. 107-111, 2004.

[28] E.M.B.-R.Y. Sun, X. Gao, and J. Kempf, "Model-Based Resource Prediction for Multihop Wireless Networks," *Proc. First IEEE Int'l Conf. Mobile Ad-Hoc and Sensor Systems (MASS '04),* pp. 114-123, Oct. 2004.

[29] S. Vural and E. Ekici, "Analysis of Hop-Distance Relationship in Spatially Random Sensor Networks," *Proc. Sixth ACM Int'l Symp. Mobile Ad Hoc Networking and Computing (ACM MobiHoc 2005),* 2005.

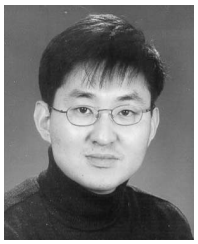
[30] DRCL J-Sim, http://www.j-sim.org, 2005.

[31] G.A. et al. "Performance Measurements of Mote Sensor Network," *Proc. ACM/IEEE Symp. Modeling, Analysis and Simulation of Wireless and Mobile System (MSWIM 2004),* Oct. 2004.

[32] O. Landsiedel, K. Wehrle, and S. Gotz, "Accurate Prediction of Power Consumption in Sensor Networks," *Proc. Second IEEE Workshop Embedded Networked Sensors (EmNetS-II),* May 2005.

[33] V. Raghunathan, C. Schurgers, S. Park, and M. Srivastava, "Energy-Aware Wireless Microsensor Networks," *IEEE Signal Processing Magazine,* vol. 19, no. 2, pp. 40-50, Mar. 2002.

Emad Felemban received the BS degree in computer engineering from King Fahd University of Petroleum and Minerals(KFUPM), Dhahran, Saudi Arabia and the MS degree from the Ohio State University, in 1998 and 2002, respectively. Currently, he is working toward the PhD degree in electrical and computer engineering, also at Ohio State University. He is working with Professor Chang-Gun Lee in real-time communication for wireless networks including ad hoc and sensor networks. He is a student member of the IEEE and the IEEE Computer Society.



Chang-Gun Lee received the BS, MS, and PhD degrees in computer engineering from Seoul National University, Korea, in 1991, 1993, and 1998, respectively. He is currently an assistant professor in the Department of Electrical Engineering, Ohio State University, Columbus. Previously, he was a research scientist in the Department of Computer Science, University of Illinois at Urbana-Champaign from March 2000 to July 2002 and a research engineer in the Advanced Telecomm. Research Lab., LG Information and Communications, Ltd. from March 1998 to February 2000. His current research interests include real-time systems, complex embedded systems, QoS management, and wireless ad-hoc networks. He is a member of the IEEE and the IEEE Computer Society.

Eylem Ekici received the BS and MS degrees in computer engineering from Bogazici University, Istanbul, Turkey, in 1997 and 1998, respectively. He received the PhD degree in electrical and computer engineering from Georgia Institute of Technology, Atlanta, in 2002. Currently, he is an assistant professor in the Department of Electrical and Computer Engineering of the Ohio State University, Columbus. His current research interests include wireless sensor networks, next generation wireless systems, and vehicular communication systems, with a focus on routing and medium access control protocols, resource management, and analysis of network architectures and protocols. He also conducts research on interfacing of dissimilar networks. He is a member of the IEEE.