Chapter 5

Real Time Scheduling
Another considerable requirement in a sensor networks is reliability, which involves techniques to guarantee data delivery to the destination successfully. In many sensor network applications, such as real-time surveillance and monitoring, the sensed data need to be delivered with some real-time constraints, such as end-to-end deadlines. Due to harsh environmental conditions, and unstable wireless communications, packet drop is very natural. Real-time data dissemination is a service of great interest to many sensor network applications. The primary contribution of this chapter is a new approach, Real Time Scheduling (RTS), which uses virtual nodes for self-stabilization in case of node failure. This approach addresses many of the shortcomings of the existing solutions. RTS delays packets at intermediate hops (not just prioritizes them) for a duration that is a function of their deadline. Delaying packets allows the network to avoid hot spotting while maintaining deadline-faithfulness. To the best of our knowledge, none of the previous work has considered delay as a parameter for real-time dissemination. Secondly, in RTS, we are proposing the use of virtual nodes for self-stabilization, that is, if any of the nodes runs out of power, instead of waiting and increasing the delay, an alternate route is selected for the transmission. The third contribution of this chapter is to explore the role of the routing protocol in the success of real-time scheduling in sensor networks. We discover that it plays a critical role that in some instances outweighs the effect of the scheduling algorithm. The basic RTS algorithm distributes the slack time (available time before the deadline expires) uniformly across all the hops. However, in a gathering data collection pattern, the amount of contention is typically higher the closer to the sink. We demonstrate the performance of RTS in comparison to state of the art models in this area (the RAP and SPEED models [34] [35]) using simulation. RTS is proposed as the real time dissemination mechanism for differentiated packet scheduling layer of the proposed architecture.
5.1 INTUITION AND MOTIVATION

In a light traffic environment, data packets are forwarded along the path from the source to sink quickly. The main contribution of the end-to-end travelling time is the processing and lower layer transmission delay. In the light traffic environment, any scheduling algorithm may not help the end-to-end performance much. For the real-time data dissemination, even routing protocol without scheduling may meet the end-to-end real-time constraints at most times.

In heavy traffic environments, large queuing delays may be experienced at intermediate nodes before they are forwarded to their next hop. This situation may occur at several intermediate nodes before the packet reaches its destination. The design of the real-time scheduling algorithm should not ignore this part of time contribution. Unfortunately, existing protocols, such as SPEED [35], RAP [34] and MMSPEED [36] all do not account for this component of the delay directly.

In traditional multi-hop communication systems, any packet in a forwarding queue would be forwarded as soon as there is no other packet with higher priority waiting to be forwarded. Traditional scheduling algorithms do not delay the packets. Given that packet transmission indirectly affects other nodes which may have urgent packets (by making the medium busy and delaying their transmission), we question whether immediately transmitting packets at each intermediate node results in a globally optimal behavior. To answer this question, we designed the Real Time Scheduling (RTS).

RTS delays packets at every hop for duration of time which is a function of the number of hops to the sink and the deadline. RTS uses an estimate of the MAC layer transmission delay and accounts for it when deciding how long to delay a packet. By delaying the packets, rather than prioritizing them, RTS achieves the following advantages:

1. A full estimate of the delay is used, including the delay due to queuing in the network layer;
2. The load is distributed over the available deadline time, potentially allowing the network to tolerate transient periods of high contention gracefully, and to avoid transient hot-spotting;

3. It provides packets with a longer chance to wait for correlated packets for purposes of aggregation or packet combining;

4. It works well with different types of routing protocols, state full or stateless, providing better performance by using shortest path (hop-count) routing, without any change of the existing popular MAC layer or lower layers protocols.

5.2 LIMITATIONS OF EXISTING SOLUTIONS

A primary challenge in real-time sensor network applications is how to carry out sensor data dissemination given source-to-sink end-to-end deadlines when the communication resources are scarce. Although routing/data transport solutions have been proposed in the context of wireless ad hoc networks, the characteristics of sensor networks make the problem different. The traffic patterns in sensor networks in response to queries or events are different from the point-to-point communication typical of sensor networks. Moreover, the bursty nature of traffic in sensor networks, as the degree of observed activity varies, can cause the network resources to be exceeded. In addition, the ad hoc nature of multi-hop sensor networks makes it difficult to schedule network traffic centrally as in traditional real-time applications.

One of the proposed solutions for real-time data dissemination [37] prioritizes packet transmission at the MAC layer according to the deadline and distance from the sink. This work has several limitations: (1) While packets are prioritized, they are not delayed when traffic is bursty, high contention results, increasing transmission and queuing delays. Furthermore, packets generated by different sensors at the same time (e.g., in response to a detected event), can lead to high collision rates. Jittering such packets can help reduce this hot-spotting; (2) MAC level solutions cannot account for the queuing delay in the routing layer (which occurs above the MAC layer); these delays can have a significant impact on end-to-end delay especially under high load; and (3) MAC level solutions require reengineering of the sensor radio hardware and
firmware, making deployment difficult and potentially causing interoperability problems with earlier hardware that supports different MAC protocols. Since the scheduling needs to consider the queuing delay in the routing layer which is above the MAC layer, the impact of the routing protocols used must be carefully examined. The effect of the routing protocol on the real-time scheduling success is not sufficiently understood. Some existing solutions [34] [35] [36] for routing in real-time traffic context provide non-deterministic routing as an extension of stateless geographic-based routing protocols. More specifically, these approaches use the best next hop with respect to the traffic/congestion situations, not only the geographic proximity as per the greedy Geographical Forwarding protocol. In addition, Geographical Forwarding, which is used in these solutions, does not always lead to the shortest delay paths, making it more difficult to meet the deadline. Furthermore, when using a longer path in terms of number of hops, increased contention for the medium results as more transmissions are needed to forward a packet.

5.3 RTS USING VIRTUAL NODES FOR SELF-STABILIZATION: BASIC ALGORITHM

The first distinguishing feature of RTS is that it considers all components of delay, including queuing delay at each forwarding node. The proposed scheme takes care of on demand routing along with a new concept of virtual nodes with power factor. In addition, RTS delays data packet transmission during forwarding for a duration that correlates with their remaining deadline and distance to the destination. Intuitively, this helps in heavy-traffic communication environment by making sure that priority inversion does not occur due to a node with only low priority packets sending and preventing a node with high priority packets from doing so. The virtual nodes help in reconstruction phase in fast selection of new routes. Selection of virtual nodes is made upon availability of nodes and battery status. Each route table has an entry for number of virtual nodes attached to it and their battery status. The algorithm [38] has been divided into three phases. Route Request (RReq), Route Repair (RRpr) and Error Phase (Err). Moreover, delaying the data packets before reaching the sink also helps the data aggregation/fusion and therefore energy efficiency; we do not explore this effect in this paper. Before a data packet reaches the sink, the end-to-end transmission
and processing delay cannot be obtained. Therefore, we use previous measurements of delay to estimate the overall delay; we call this estimate the End-to-End Estimate of Transmission Delay (EETD) [35]. The one hop estimate is called ETD. Summing the ETDs of a data packet hop by hop during its forwarding can lead to inaccurate estimates since one hop ETD can fluctuate significantly. Therefore, we use the following function to decide the EETD:

\[
EETD = ETD \cdot \frac{E2E Distance}{OHD}
\]  

5.1

Where OHD is One Hop Distance and the distance can be measured in different ways.

Different RTS scheduling policies can be developed based on the allocation of the available slack time among the different hops. The target transmission times are either set by the source or computed at intermediate hops based on a known algorithm. In the base RTS algorithm, the target transmission time is set to be equal at all hops and is determined as follows:

\[
TD = \frac{DL - EETD}{Distance(X, Sink)} \cdot \alpha
\]  

5.2

Where TD be the transmission delay, DL be the deadline and the \( \alpha \) is a constant “safety” factor for insurance that the real-time deadline would be met. For example, setting \( \alpha \) to be 0.7, would target delaying the packet 70% of the available slack time, leaving the remaining time as a safety margin.

As we can see, the Target Delay of any in-queue packet determines its priority. The time a packet is delayed in the queue can be used as the key to a priority queue that holds the packets to be transmitted. The end-to-end transmission and processing delay is considered along with the queuing delay, by taking into account the end-to-end deadline, distance and EETD.

We consider static vs. dynamic versions of the protocols depending on whether the target transmission times are set by the source and followed by intermediate nodes (static), or whether they are computed/adjusted at intermediate nodes (dynamic).
5.3.1 Static Real Time Scheduling (SRTS)

In static RTS, the target delay is set with the values of parameters at the data source. In the equation 5.2, the end-to-end deadline is fixed at the data source; the EETD is measured with the ETD of forwarding node and the distance from source to sink (X is the data source). So even we call it static, the different ETDs of forwarding nodes would make the target delay at each node different.

5.3.2 Dynamic Real Time Scheduling (DRTS)

In dynamic RTS, the target delay is reset at each forwarding node with the local value of parameters. In equation 5.2, the end-to-end deadline of a packet at some forwarding node is the remaining slack time, measured by E2E Deadline-Elapsed Time. The EETD is decided by the one-hop ETD of the forwarding node and the distance from it to the sink, not the distance from source to sink. So the dynamic RTS is able to continuously refine the priority of the packet.

5.3.3 Non-linear Real Time Scheduling (NLRTS)

It is also possible to allocate the available slack time non-uniformly among the intermediate hops along the path to the sink. For example, we may desire to provide the packets with additional time as it gets closer to the sink. The intuition is that in a gathering application, the contention is higher as the packet moves closer to the sink. Different policies can be developed to break down the available time. We explore the following policy:

\[ TD = \frac{E2E\ deadline - EETD}{\frac{RD}{2OHD}} \cdot \alpha \quad 5.3 \]

Where RD is remaining distance and OHD is one hop distance.

More generally, we may want to allocate the slack time proportionately to the degree of contention along the path. Such a heuristic may be developed by passing the contention information along with the routing advertisement and allocating the available slack time accordingly. Finally, one may decide to favor aggregation by delaying packets closer to the source where the data is more correlated.
5.4 RTS IMPLEMENTATION

RTS does not ignore the queuing delay. It considers both the transmission delay and the queuing delay by doing a set of very simple scheduling decisions. The basic RTS scheduling algorithm has been shown in Section 5.3. But RTS is more than that. Although the term RTS stands for Real-Time Scheduling, it is not only a scheduling algorithm. It involves the architecture design of the whole system. The typical architecture of a system that RTS works on is shown in Figure 5.1. The RTS scheduler resides above (or within) the routing layer. It uses routing level information, such as the end-to-end distance, in making its scheduling decisions. For any real-time applications based on sensor networks, the end-to-end real-time deadline is assumed to be included on the data packet itself. Figure 5.1 shows an example of how this information is collected. While, in this figure, the MAC layer is shown, the RTS scheduler and the MAC layer protocol are not aware of each other.

The scheme of virtual nodes has been explained with the help of an example shown in Figure 5.2. Assume that the node A is the source while destination is the node D. Note that the route discovered using new scheme routing protocol may not necessarily be the shortest route between a source destination pair. If the node C is having power status in critical or danger zone, then though the shortest path is A-B-C-D but the more stable path A-B-H-G-F-E-D, in terms of active power status, is chosen.

Figure 5.1: RTS Architecture

This may lead to slight delay but improves overall efficiency of the protocol by sending more packets without link break than the state when some node is unable to
process route due to inadequate battery power. The process may help when some intermediate node moves out of the range and link break occurs; in that case virtual nodes take care of the process and the route is established again without much overhead.

Figure 5.1: An example of routing

In Figure 5.2, if the node G moves out, the new established route will be A-B-H-I-F-E-D. Here, the node I is acting as virtual node (VN) for the node H and the node G. Similarly, the node J can be VN for the nodes D, E, K. Virtual node (VN) has been selected at one hop distance from the said node. In this scheme, the virtual nodes help in reconstruction phase in fast selection of new routes. Selection of virtual nodes is made upon availability of nodes and battery status. Each route table has an entry for number of virtual nodes attached to it and their battery status. The virtual node scheme is divided into three phases. Route Request (RReq), Route Repair (RRpr) and Error Phase (Err).

5.4.1 Route Construction (RReq) Phase

This scheme can be incorporated with reactive routing protocols that build routes on demand via a query and reply procedure. The scheme does not require any modification to the QDPRA [39] RReq (route request) propagation process. In this scheme, when a source needs to initiate a data session to a destination but does not have any route information, it searches a route by flooding a ROUTE REQUEST (RReq) packet. Each RReq packet has a unique identifier so that nodes can detect and drop duplicate packets. An Intermediate node with an active route (in terms of power
and Virtual Nodes), upon receiving a no duplicate RReq, records the previous hop and
the source node information in its route table, i.e., backward learning. It then
broadcasts the packet or sends back a ROUTE REPLY (RRep) packet to the source if
it has an active route to the destination. The destination node sends aRRep via the
selected route when it receives the first RReq or subsequent RReqs that traversed a
better active route. Nodes monitor the link status of next hops in active routes. When
a link break in an active route is detected, an Err message is used to notify that the
loss of link has occurred to its one hop neighbor. Here, Err message indicates those
destinations which are no longer reachable. Taking advantage of the broadcast nature
of wireless communications, a node promiscuously overhears packets that are
transmitted by their neighboring nodes. When a node that is not part of the route
overhears a RRpr packet not directed to itself transmitted by a neighbor (on the
primary route), it records that neighbor as the next hop to the destination in its
alternate route table. From these packets, a node obtains alternate path information
and makes entries of these virtual nodes (VN) in its route table. If route breaks occurs,
it just starts route construction phase from that node. The protocol updates list of VNs
and their power status periodically in the route table.

5.4.2 Route Error & Maintenance
In this scheme, data transmits continuously through the primary route unless there is a
route disconnection. When a node detects a link break, it performs a one hop data
broadcast to its immediate neighbors. The node specifies in the data header that the
link is disconnected and thus the packet is candidate for alternate routing. Upon
receiving this packet, route maintenance phase starts by selecting alternate path and
checking power status.

5.4.3 Local Route Repair (Err Phase)
When a link break in an active route occurs, the node upstream of that break may
choose to repair the link locally if the destination was no farther and there exists VNs
that are active. The Time to live (TTL) of the RReq should initially be set to the
following value:

\[ TTL = \max (\text{Min}_{\text{Rpr}}\_TTL + \text{VN}, 0.5 \times \#\text{hops}) + \text{power status} \]
Where Min_Rpr_TTL is the last known hop count to destination, #hops is the number of hops to the sender of the currently undeliverable packets. VN is the virtual nodes attached to the said node and the power status is power state of the node at that time. As $0.5 \times #hops$ is always less than Min_Rpr_TTL + VN, so the whole process becomes invisible to the originating node.

This factor is transmitted to all the nodes to select best available path with maximum power.

Figure 5.3 gives an idea of working of local route repair. Initial path from source node A to destination node X is shown via solid lines. When link breaks at node N, route repair starts, node N starts searching for new paths, buffering packets from A-B in its buffer. Node N invokes Route Request phase for X. Now backbone nodes are selected and proper selection of nodes is done based on power factor. Path selected becomes [N-L-M-K-X], instead of [N-L-P-X], since the node P is not in active state. Even though the route may become longer, but the selected route path is far more stable and delivers more packets. Stability of route depends upon two major aspects as: Life time and Power status. The concept has been explained in Table 5.1.
TABLE 5.1: Active Time Estimation

<table>
<thead>
<tr>
<th>Node</th>
<th>VN</th>
<th>Min_TTL</th>
<th>#hops*0.5</th>
<th>Power Status</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>3</td>
<td>3</td>
<td>1/2</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>M</td>
<td>4</td>
<td>2</td>
<td>2/2</td>
<td>8.5</td>
<td>14.5</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>1</td>
<td>3/2</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>P</td>
<td>3</td>
<td>1</td>
<td>2/2</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>Q</td>
<td>3</td>
<td>1</td>
<td>3/2</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>P1</td>
<td>1</td>
<td>4</td>
<td>1/2</td>
<td>7</td>
<td>12.5</td>
</tr>
<tr>
<td>P2</td>
<td>2</td>
<td>3</td>
<td>2/2</td>
<td>7</td>
<td>12</td>
</tr>
<tr>
<td>G1</td>
<td>2</td>
<td>1</td>
<td>4/2</td>
<td>7.5</td>
<td>10.5</td>
</tr>
<tr>
<td>L1</td>
<td>1</td>
<td>3</td>
<td>2/2</td>
<td>8.0</td>
<td>12</td>
</tr>
</tbody>
</table>

When selection has to be made between nodes P1 and L at the start of repair phase, selection of node L has the advantage over node P1. Similarly, in the selection between nodes K and K1, node K has higher weight. If any VN has not been on active scale, it is rejected and a new node is searched. In addition to power factor, efforts are made to keep the path shortest. This local repair attempts are often invisible to the originating node. During local repair data packets will be buffered at local originator. If, at the end of the discovery period, the repairing node has not received a reply message RRpr, it proceeds in by transmitting a route error Err to the originating node. On the other hand, if the node receives one or more route reply RRep’s during the discovery period, it first compares the hop count of the new route with the value in the hop count field of the invalid route table entry for that destination. Repairing the link locally is likely to increase the number of data packets that are able to be delivered to the destinations, since data packets will not be dropped as the ERR travels to the originating node. Sending a ERR to the originating node after locally repairing the link break may allow the originator to find a fresh route to the destination that is better, based on current node positions. However, it does not require the originating node to rebuild the route, as the originator may be done, or nearly done, with the data session. In AODV, a route is timed out when it is not used and updated for certain duration of time. The scheme uses the same technique for timing out alternate routes.
5.5 RTS FOR DIFFERENT ROUTING PROTOCOLS

Main advantage of RTS is that it can be adapted to any underlying routing protocol. However, the RTS algorithm may need to be adapted to consider the cost metric used by the routing algorithm. For example, in a system based on the shortest path routing (SP), the distance parameters used by RTS scheduler is measured in number of hops. The corresponding functions are:

\[
EETD = ETD \cdot E2E \text{ hops} \tag{5.5}
\]

\[
TD = \frac{\text{Deadline} - EETD}{H} \cdot \alpha \tag{5.6}
\]

\[
TD = \frac{E2E \text{ Deadline} - EETD}{2H} \cdot \alpha \tag{5.7}
\]

Where H stands for number of end-to-end hops. For the geometric routing, the values of distance parameters used in RTS Scheduler would be the Euclidean distance. In summary, the following information is needed to schedule packets in RTS:

- End-to-end deadline information: This information is provided by the application in the data packet as required by any real-time data dissemination application. For those applications where the header of data packet does not include this information, an alternative way for RTS to obtain the end-to-end deadline information is needed.

- End-to-end distance information: This information is obtained from the routing protocol. For example, this information is maintained in the routing tables of traditional distance vector based or link-state based routing protocols to keep track of the cost of the path. Furthermore, in source routed protocols, such as DSR, this information can be directly computed from the packet header which includes the full path to the destination. Finally, in geographic routing, Euclidian distance measured as the distance from the current node to the destination can be used as the distance metric. The output of RTS scheduler is the queuing delay, which is used by the routing protocol to decide how long to delay an incoming data packet before attempting to forward it (by
passing it to the MAC layer). MAC layer prioritization is not needed by the RTS design since the packets are sent when their real time local deadline is reached; they should all be of roughly equal priority. Not requiring changes to the MAC layer is a desirable feature of RTS relative to RAP.

5.6 PROPERTIES OF RTS

In summary, following are the design features of the RTS framework:

- Ability to interoperate with different routing protocols: Unlike the SPEED [35] or RAP [34] framework, which are specific to geographical routing, RTS is not limited to a specific routing protocol. Instead, it can operate directly with any hop-based cost metric protocol and can be easily adapted to work with Geometric routing protocols.

- This scheme utilizes a mesh structure and alternate paths in case of failure. The scheme can be incorporated into any on-demand unicast routing protocol to improve reliable packet delivery in the face of node movements and route breaks. Alternate routes are utilized only when data packets cannot be delivered through the primary route. As a case study, the proposed scheme has been applied to QDPRA [39], and it was observed that the performance improved. Simulation results indicated that the technique provides robustness to mobility and enhances protocol performance. It was found that overhead in this protocol was slightly higher than others, which is due to the reason that it requires more calculation initially for checking virtual nodes.

- Soft Real-time: RTS maintains a uniform delivery speed of data packets, meeting the deadline of most data traffic with best effort. Packets that pass their deadline are not dropped. While it is possible to better support hard real-time in this framework (for example, by increasing the safety margin, and immediately dropping packets that are late), we do not pursue such extensions.

- No MAC layer support required: Unlike the SPEED or RAP, RTS does not require MAC layer support for prioritized scheduling (as with RAP) or for tracking delay (as with SPEED). This makes RTS readily deployable on existing infrastructure.
• QoS routing: RTS integrates the transmission delay with the queuing delay, considering both the lower layer communication cost and that of higher layers and differentiating the data flows with different real-time constraints.

• Ability to withstand high load and hot spotting: RTS uses the queuing mechanism to delay any data flows to restrict contention to occur among only the most urgent traffic. This allows RTS to gracefully accommodate higher traffic levels than RAP or SPEED.

• Data Fusion: RTS tries to delay any incoming data traffic which gives more possibility of the data aggregation operations. Since the data aggregation is a primary data operation during the data forwarding for most applications, RTS fits better than the other approaches which attempt to send packets without delay.

5.7 EXPERIMENTAL EVALUATION OF RTS

We implemented RTS (Static, Dynamic and Non-Linear) with both the Shortest Path routing and Greedy Geographic Forwarding in the Network Simulator (J-SIM) [70]. We also implemented the RAP Velocity Monotonic Scheduling (VMS) [34] [40] with GF, including the specialized MAC support required by it on J-SIM per the specification. Since GF has been shown to significantly outperform traditional routing protocols such as DSR [10] and deadline-based scheduling, in the context of sensor network data dissemination, we restrict the routing comparison to GF and SP, and the scheduling comparison to VMS (Velocity Monotonic Scheduling) and RTS.

TABLE 5.2: Simulation Parameters of RTS

<table>
<thead>
<tr>
<th>Mac layer protocol</th>
<th>IEEE 802.11 with prioritizing extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmission Radio Range</td>
<td>250 m</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>2 Mbps</td>
</tr>
<tr>
<td>Data Packet Size</td>
<td>32 B</td>
</tr>
<tr>
<td>Data Rate</td>
<td>2 packets/second</td>
</tr>
<tr>
<td>Simulation Area</td>
<td>1000 × 1000 m²</td>
</tr>
<tr>
<td>Number of Sensor nodes</td>
<td>100</td>
</tr>
<tr>
<td>Effective Simulation Time</td>
<td>120 sec</td>
</tr>
</tbody>
</table>
Table 5.2 shows the simulation parameters we use; unless otherwise indicated these parameters are used in the studies. We use both the grid and random deployment to simulate our algorithm. In grid deployment, we divide the covered simulation area into a $10 \times 10$ grid. One of the 100 sensor nodes is placed at the center of each of the grid tiles. The sink is placed on the northwest corner of the network. Nodes publish data at the rate of 2 packets per second in order to simulate a fairly high load traffic scenario. In random deployment, the 100 nodes are randomly placed in the simulation area while the sink is placed roughly at the center of the area. First, we compared RTS with VMS both using the same routing protocol (GF); recall that GF was used in the original RAP scheme [34]. Later, we show that SP significantly outperforms GF for RTS. Since we consider soft real-time applications, a change we made to the RAP mechanism is that each node tries to forward all incoming data packets, no matter if the deadline is already missed or not. In the original implementation of RAP, the packets missing the deadline would be dropped. Since RTS does not require any MAC layer information, we use the original IEEE 802.11 as our MAC layer protocol. We considered the issue of what the RTS safety margin parameter $\alpha$ should be set to. If $\alpha$ is too high, packet delay variability can cause deadlines to be missed since most of the slack time is taken up by intentional RTS delay and unexpected delays cause a packet to miss the deadline. Conversely, if $\alpha$ is too low, packets are conservatively sent quickly towards the sink, possibly overflowing buffers around it. Experimentally, we observed that a safety margin parameter of 0.7 works well across different deadlines. Thus, 30% of the deadline budget is set aside to account for inaccuracies in ETD estimates and/or unexpected transmission or queuing delays.

The first experiment studies the performance of RTS scheduling for sensor networks relative to RAP. Figure 5.4 shows that for different packet requirement, the miss ratios and drop ratios of RTS Static and Dynamic are much lower than those of DVM and SVM for across all the considered deadline range. Static VMS (SVM) [40] prioritizes packets based on a fixed requested velocity that is computed at packet generation time. The values of parameters used by prioritizing at any intermediate node are that of the data source, not the forwarding node. Dynamic VMS (DVM) re-computes packet velocity at each intermediate node. For a packet from a source whose distance is $d$, urgency is not only based on the end-to-end deadline, but also the time is already elapsed. Dynamic RTS outperforms static RTS in terms of the miss ratio.
5.7.1 Performance under Random Deployment

RTS and VMS were also evaluated using a random deployment scenario where the 100 nodes were randomly placed within the simulation area. Three random deployments of 100 sensor nodes in a 1000 × 1000m² areas are taken. Each result represents the average of several experiments with different seeds. We varied the deadline requirements from 0.5 to 2.0 seconds in steps of 0.5 seconds. Ratios and drop ratios for the different algorithms. The simulations show that both RTS and VMS (Figure 5.5) perform much better in random scenarios than they did in the grid scenarios possibly because the location of the sink is central to the simulation area,
making the average sensor distance to the sink smaller. Again, RTS provides superior performance to VMS. For the VMS, the drop ratios do not decrease as the deadline grows since it prioritizes but does not delay packets. The drop ratio becomes the lower bound of the miss ratio. RTS shows more reactivity since both the drop ratio and miss ratio keep decreasing as the deadline requirement is relaxed.

5.7.2 Performance under Busty Traffic

In this study, we evaluate the performance of RTS vs. RAP under busty traffic conditions. Each node alternately publishes packets at the pre-set data rate for 5 seconds, then stops publishing for the second 5. Figure 5.6 shows the miss ratios and drop ratio of RTS and SVM under this busty traffic with end-to-end deadline from 0.1 second to 3.0 seconds. From the Figure 5.6, we can see that the miss ratio of dynamic RTS is much lower than that of SVM with the busty traffic, because RTS can tolerate the traffic burst by delaying some packets, and taking advantage of the idle period. On the other hand, SVM cannot make use of the traffic behavior since it does not delay packets. The decrease in the drop ratio shows that RTS also delivers more packets as the deadline constraints are relaxed.

5.7.3 Comparison with SPEED

We also built simulation models for the SPEED framework within the Java simulator. Unfortunately, the simulation results we obtain do not match the performance demonstrated in the original SPEED papers [34][35].

![Figure 5.4: Busty traffic](image-url)
We implemented the full specification of SPEED, SPEED-T (Minimal one hop delay first), and SPEED-S (maxim alone hop progress speed first), simulating them in exactly the same scenarios as specified in [34]. Our experiences with SPEED show that it performs extremely poorly at high loads because its backpressure mechanism is not suited to the situations where alternative paths are also congested. In those situations backpressure ends up increasing the load on the network by routing packets through unnecessarily long paths. We believe that our comparison is fair because all the algorithms are implemented in the same environment (thus removing any differences that occur due to the different simulators). Because of the overall poor performance under high load, we do not compare RTS with SPEED in detail.

5.8 CONCLUSIONS

Real-time data dissemination is a service of great interest to many sensor network applications. This chapter proposed and evaluated the Real Time Scheduling using virtual nodes for self-stabilization mechanism for real-time sensor network applications. RTS offers significant advantages over existing real-time sensor data dissemination schemes. It accomplishes real-time support by delaying packets a fraction of their slack time at each hop. As a result, it is better able to tolerate bursts than schemes that simply prioritize packet transmission. RTS can operate with simple routing protocols easily and outperforms existing solutions in both the miss ratio and overall delay. RTS is a network layer solution and does not require changes to lower level protocols making it easier to deploy and independent of the underlying sensor network hardware capabilities. Virtual nodes support self-stabilization, that is, if any node goes out of power without waiting for node repair, we will go for alternate path; this will decrease idle delay. Using simulation, we found the drop ratio is the lower bound of the miss ratio of real-time communication. If the drop ratio is decreased, given a reasonable end-to-end deadline, the miss ratio of these real-time applications should also be decreased. Mostly the packets are dropped due