

Adaptive Resource Control Scheme to Alleviate Congestion in Sensor Networks

Jaewon Kang, Badri Nath
DATAMAN Lab.
Rutgers University
{jwkang,badri}@cs.rutgers.edu

Yanyong Zhang, Shengchao Yu
WINLAB
Rutgers University
{yyzhang,yusc}@ece.rutgers.edu

Abstract

Sensor networks are being increasingly deployed for surveillance and monitoring applications. These networks will suffer from severe congestion as soon as the target events occur. During congestion, important data packets may be dropped, which can essentially nullify the purpose of sensor networks.

In this paper, we propose an adaptive resource control scheme to alleviate congestion in sensor networks. The choice of resource control is necessitated by the following two factors: (1) unlike in traditional networks, we cannot simply reduce the source traffic because these data are critical to the applications; and (2) sensor networks usually are densely deployed and thus have the capability of provisioning more resources when needed. The proposed scheme adjusts the resource provisioning based on the congestion level so that we can both increase the capacity, by having more sensor nodes forwarding data or having more routing paths, to alleviate the congestion, as well as reduce the capacity after the congestion to conserve the energy consumption.

1. Introduction

The recent advancements in MEMS technology, processor design, and wireless communication have enabled a wide range of monitoring applications using networks of sensor nodes. To name just a few examples, one can deploy a sensor network to study the behavior of an endangered species, to report fires in a forest, to monitor the health of a building, or to collect the traffic information on a busy highway, etc. These networks will usually be left unattended, with each individual node sensing the physical environment, performing processing if necessary, and reporting the data to the sinks through multi-hop wireless communication. In order to conserve energy which is the most scarce resource, a sensor network requires low reporting rates from the source nodes during the periods when the specific events

are not observed. However, a much higher rate will be needed as soon as these events occur such that necessary actions can be taken promptly. For example, a temperature sensor as part of the fire detection application only needs to report 1 packet every 5 minutes when its reading is below 50 degree, but it must report more than 100 packets per minute once its readings exceed 100 degree [12]. As a result, sensor networks experience traffic alternating between periods with a very low traffic volume (referred to as *dormant state*) and periods with a high traffic volume (referred to as *crisis state*).

Once in a crisis state, the sudden traffic increase may lead to congestion in the network [12, 16]. When congestion occurs, the network will enter into an unstable state and packets will be randomly dropped. This is particularly undesirable because the data generated during the crisis state are of great importance, often critical, to the applications.

This paper thus focuses on *alleviating congestion* in sensor networks. Upon detection, we have two options to control the congestion: throttling the source traffic volume (referred to as *traffic control*), and increasing the network resource (referred to as *resource control*). Although traffic control strategies are effective in traditional wired networks and are also suggested in some sensor network scenarios [12, 16], they are unsuitable for our purpose for the following three reasons. First, as a result of conserving energy, only a small number of sources will be reporting an event, and these data will be of great value to the applications. Failure to ship these data to the sink means a substantial loss to the applications. Second, it is highly likely that the traffic bursts are transient. For example, sensor nodes that monitor an animal will generate transient bursts of traffic when the animal moves around quickly. It is also possible that an incorrect reading of a sensor node can generate a transient burst since these devices have poor reliability. For these very short-term congestion, it is not efficient to reduce the source traffic due to its high overhead. Third, there is usually an abundance of resources in sensor networks (unlike its wired or other wireless counterparts) because these networks are usually densely deployed in order to achieve

a reasonable network lifetime [1, 19, 18]. A natural way of using the resource is to conserve as much as possible during a dormant state while using them wisely during crises.

This paper investigates a framework to alleviate congestion by turning on more resources as soon as the congestion is detected. In great detail it examines a family of strategies of managing the network resource during a crisis state. These strategies include the following features:

- *Congestion alleviation.* In this paper, we investigate the approach of using adaptive resource control strategies to alleviate congestion. We suggest that more routing paths need to be set up in order to share the load after the flow level congestion is detected. Sometimes, we need to wake up the sleeping nodes (these nodes were in sleep state to conserve energy) to form new routing paths. It is particularly important to control the resources at different temporal and spatial granularities.
- *Energy awareness.* While the long-term resource control scheme can reduce the congestion within the network, it also significantly increases the energy consumption. As a result, we would like to turn off the extra resources as soon as the source traffic decreases. Our scheme achieves this because we can detect the congestion change within the network timely.

To our knowledge, this is the first attempt to guarantee data delivery even under severe sensor network congestion by increasing the resource provisioning that participates in the data delivery. In addition, the proposed algorithm reduces the resource provisioning after congestion subsides. We have shown that our resource control scheme can cut down the packet drop ratio while consuming less energy than the case without resource control.

The rest of the sections are organized as follows. Section 2 reviews the related work. Section 3 presents our proposed adaptive resource control scheme. Section 4 describes simulation results. Section 5 concludes the paper.

2. Related Work

Even though congestion control in sensor networks is important not only to make sure that the reported samples don't exceed the network capacity but also to optimize the lifetime of the network by saving scarce energy in sensor nodes, it has received attention only recently [16].

A guideline of congestion control in sensor networks was given by [13]. They suggest that the congestion control must not only be based on the network capacity, but also on the accuracy level required by the observer, i.e. application, at the sink node. In other words, the total data received from M reporting sensors represented as $\sum_{i=1}^M b(S_i)$, where

$b(S_i)$ is the bit rate of sensor i , should not exceed a certain fraction of the available channel capacity C_{total} , but also be high enough to satisfy the desired accuracy as shown in Equation 1. $C_{application}$ is the application-specific accuracy level.

$$C_{application} \leq \sum_{i=1}^M b(S_i) \leq \alpha C_{total} \quad (1)$$

We can infer from Equation 1 that if the observer requires a high accuracy *temporarily* during a crisis state (i.e. $C_{application}$ exceeds $\sum_{i=1}^M b(S_i)$) and $\sum_{i=1}^M b(S_i)$ cannot be increased due to the upper bound imposed by αC_{total} , then the available channel capacity C_{total} should be temporarily increased to allow the increased $\sum_{i=1}^M b(S_i)$, therefore to meet the accuracy requirement in Equation 1.

In [12], they propose a reliable transport technique called event-to-sink reliable transport (ESRT) protocol. In ESRT, the sink periodically configures the source sending rates to avoid congestion. They observe that there will be many sources that are monitoring the same event and sending the data back to the sink. Congestion will occur when these sources send data faster than a certain threshold. The sink will detect congestion using two observations: (1) when the number of received packets within a window is below the desired level; and (2) when an intermediate node reports possible buffer queue overflow by marking 1-bit CN field in the header of the delivered packet(s). In ESRT, however, flows are not differentiated and sending rates of all flows are throttled when congestion is detected.

In CODA [16], they presents a detailed study on congestion avoidance in sensor networks. The basic idea is that as soon as congestion occurs, the source (or an intermediate node)'s sending rates must be reduced to quickly release the congestion. In the simple case, as soon as a node detects congestion, it broadcasts a backpressure message upstream. An upstream node that receives the backpressure can decide to drop packets, preventing its queue from building up and thus controlling congestion. If multiple sources are sending packets to a sink, CODA also provides a method of asserting congestion control over these multiple sources by requiring constant feedback (ACKs) from the sinks. If a source does not receive the ACKs at predefined times, it will start throttling the sending rates.

Most studies on congestion control in sensor networks in the literature deals with traffic-controlling aspect of congestion control, i.e. they basically try to reduce the incoming traffic into the network during congestion. However, we believe that when congestion is transient or reducing traffic hurts the accuracy level significantly, increasing resources such as hop-by-hop bandwidth, additional data path can be a better option to cope with congestion.

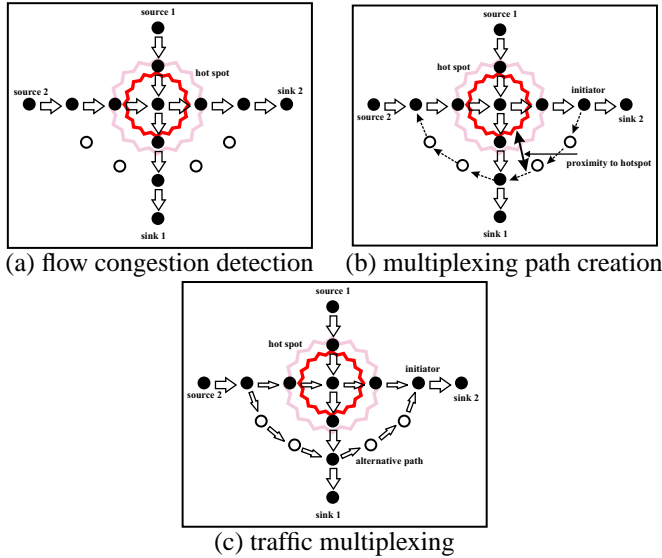


Figure 1. Schematic of creating a multiplexing path

3. Resource Control Strategies to Alleviate Congestion

Three factors motivate the usage of resource control schemes in sensor networks. First, reducing source traffic during a crisis state is undesirable since it reduces the accuracy level observed by applications. Second, congestion in sensor networks is often transient by nature which can be alleviated by quickly adjusting the network resource provisioning. Third, sensor networks usually have slack capacity to be turned on when needed.

Our resource control scheme has two phases: (1) to increase resource provisioning as soon as congestion occurs; and (2) to reduce the resource budget as soon as congestion subsides. Our scheme has managed to balance the need of reliably shipping data packets to the sink during a crisis state and the need of conserving energy consumption.

3.1. Basic Idea

During a dormant state, the sensor network usually turns off a large number of nodes to extend the network lifetime [23, 20, 22, 14]. As soon as flow-level congestion is detected, we propose to incorporate these nodes into routing by forming one or more additional paths called *multiplexing paths*, and distributing the incoming traffic over the original path and the multiplexing paths.

The algorithm involves three steps:

- *Congestion Detection* The upstream nodes of a flow periodically notify their congestion level to down-

stream nodes by embedding their perceived congestion level into the header of data packets. After detecting the flow-level congestion exceeds a predefined congestion threshold, the first node whose congestion level is below the *hotspot proximity threshold* will become the *initiator* and start creating multiplexing paths.

- *Alternative Path Creation* (Figure 1(b)). Building multiplexing paths is a challenging task. If the path is too close to the hot spot, it does not help, but it will further interfere with the already bad congestion. On the other hand, if the path is too far from the original one, then the packet delivery latency and the energy consumption will increase. In addition, building multiplexing path can be made even more complicated by other factors. For example, one may not be able to find a path from the initiator to the source; or there may exist another hot spot within the network. In this paper, we propose to solve the aforementioned dilemma by examining each candidate node's congestion level.
- *Traffic Multiplexing* (Figure 1(c)). The node(s) where the multiplexing paths meet the original path are referred to as *traffic dispatcher*. The dispatcher will evenly distribute the traffic between multiple paths in a round robin fashion so that the congestion on the original path can be alleviated.

The three steps illustrated in Figure 1 correspond to the resource expansion phase once congestion occurs. We also have a resource shrinking phase after the transient congestion ends, and it is discussed in Section 3.4.

In the following sections, we discuss these three steps in great deal and the numerous design issues such as

- How a node gets the congestion levels of neighbor nodes ?
- What is the criteria to choose the next-hop neighbor given the list of congestion levels of its neighbors ?
- How a node decides the next-hop neighbor so that the multiplexing path expands towards the source ?

3.2. Alternative Path Selection

In this section, we discuss how a multiplexing path can be established. We first discuss the mechanism to make sleeping nodes join the routing process. We then discuss the necessary data structure for the algorithm. Finally we discuss the design issues that are involved.

3.2.1 Nodes Wake-up

As shown in [23, 20, 22, 14], it is a common practice in sensor networks to make extra nodes, referred to as *backup*

Table 1. An example neighbor table

neighbor ID	congestion level	remaining energy (J)	proximity to flows
12	0.9	0.2	2 (f1), 4 (f2)
2	0.3	0.01	4 (f1), 1 (f3)
23	0.4	30	5 (f1)
34	0.2	20	3 (f1)

nodes, into sleep mode (by turning off their radio) so that the overall resource (e.g., energy) is saved during a dormant state. However, these nodes need to wake up periodically to check whether they are needed, such as when a node on the routing path runs out of battery. The time between two subsequent wake-ups is referred to as the *sleep interval*. The choice of the sleep interval is of great importance because it is undesirable to wake up too frequently or too infrequently. Numerous studies have focused on how to calculate an optimal sleep interval. In order to make the backup nodes congestion-conscious, we propose to adapt the sleep interval based on their congestion level.

Every time when a backup node wakes up, it measures its congestion level. If it observes that current congestion level is greater than the previous measurement, it will shorten its sleep interval and wake up more often. As soon as its local congestion level is above a threshold, it will significantly reduce its sleep interval and becomes “alert”.

During a dormant state, the congestion level around a backup node is extremely low because no traffic is routed through them. As soon as there is a hot spot, the backup nodes around the hot spot will first become active. Once they become active, they communicate with each other (described in Section 3.2.2), and the generated traffic will increase the congestion of those backup nodes that are nearby. Similarly, a large number of backup nodes that are not too far away from the hot spot can thus be woken up and become ready to be picked to form multiplexing paths.

3.2.2 Neighbor Table

Our algorithm is independent from the underlying routing protocol. The only requirement it has is that every node within the network should maintain a *neighbor table* which contains information about all its neighboring nodes. This requirement can be easily met in virtually any routing protocol, including plain flooding protocol. Simple periodic broadcasting suffices for this purpose.

An example neighbor table is shown in Table 1. Each neighbor entry in the table contains the neighbor ID, the congestion level, the remaining energy level, and the proximity to nearby flows. The congestion level and remaining energy of a neighbor are obtained by overhearing the data packets from the neighbor. For this, each node embeds

these information into the data packet header periodically or when they change a lot.

The proximity to flows field in the neighbor table helps to choose the neighbor which is closer to the original path during multiplexing path selection. To construct this, when a original path is established, the nodes along the path broadcasts a dummy message with the flow ID, which is unique for each source-sink pair and set by a source, to their neighbors. The neighbors increase the hop count in the header of the message and broadcast again. The dummy message does not propagate more than m hops. The neighbors will update their proximity to a certain flow based on the hop count in the header of the message. Each node periodically broadcasts its flow ID and hops to the flow to its neighbors.

3.2.3 Congestion- and Energy-aware Path Selection

Creating multiple paths has been extensively studied for many purposes such as reliability, load balancing, security, etc [10, 15, 4, 17, 9, 11, 2, 3, 6, 7, 21]. The difference between our scheme and these protocols is that a multiplexing path is created mainly based the congestion level dynamically by the intermediate node. In detail, our multiplexing path selection is based on each candidate node’s congestion level, energy level, and proximity to the flow. In order to explain our algorithm, let us try to answer the following question: if a node has 4 neighbors that are the candidates for the next hop in the multiplexing path as shown in Table 1, which candidate should it choose ?

As mentioned earlier, each candidate has three parameters: congestion level, energy level, and proximity to the flow. Our decision will thus be based on the interplay of these three parameters. Between these three parameters, we think congestion level is more important because we want to make sure that the candidate node itself is not already highly congested. The other reason is that we expect many congestions are only transient. Hence, we first sort all the candidates in the ascending order of their congestion level. We will not consider those whose congestion level is already above the hotspot proximity threshold. If we set the hotspot proximity threshold at 0.5, then node 12 from Table 1 will not be considered. Among all the nodes that have a low congestion measurement, we further remove those nodes that do not have enough energy. As a result, node 2 will not be considered even though its congestion level is below the threshold. We use the term *qualified* candidate nodes to denote all the nodes whose congestion level is below the hotspot proximity threshold and energy level is above a certain threshold. Another rule to make sure that the multiplexing path is merged with the original path is that among all these qualified candidate nodes, we propose to choose the one that has the lowest proximity to the flow instead of those whose congestion level is the lowest or pos-

sibly 0. A congestion level being close to 0 implies: (1) the candidate node is probably far away from the original path and it can be difficult to find a multiplexing path that intersects with the original one; and (2) the backup nodes around this candidate node may be still sleeping, and it is difficult to establish a path. Therefore, if the corresponding flow is flow 1, then node 34 whose proximity to flow 1 is smaller is chosen for next hop.

If a node does not have any qualified candidate neighboring node, it must stop the searching and send the negative feedback to the node that selected it as the candidate. After its precedent node receives the feedback, it will mark the failed node in its neighbor table and choose another candidate node. The multiplexing path selection works like searching in a depth-first search tree. Finally, if the initiator on the original path receives a negative feedback from the candidate node, it asks the next node on the original routing path to become the initiator and starts the algorithm.

Figure 2 depicts the schematic of a multiplexing path selection. (a) depicts two cross traffic (one flow from node 1 to node 13 and the other flow from node 5 to node 9) create a hotspot around node 6. The nodes connected by the dotted line are within each other's radio range. In (b), the highest per-node congestion level (1.2 of node 3) is forwarded to the downstream nodes. If the hotspot proximity threshold is set to 0.5, then node 6 and 10 cannot be an initiator. Node 12 whose congestion level is less than the hotspot proximity threshold starts a multiplexing path selection procedure by contacting node 11 that is not on the original path in (c). Node 11 has two neighbors that are not on the original path, node 7 and 8. If the congestion levels of both nodes are less than the hotspot proximity threshold, then the node that is close to the original path will be chosen for next hop. In the figure, node 7's congestion level (0.7) exceeds the hotspot proximity threshold (0.5). Node 7's congestion level is high because it is near the hotspot. Therefore, node 8 is selected for next hop. After the multiplexing path is established, the incoming traffic is multiplexed onto the original data path and the multiplexing data path by node 2 acting as a traffic dispatcher.

The initiator (node 12) keeps scanning the congestion level of the flow from both paths by inspecting the per-flow congestion level recorded in the data packets from two paths. If the per-flow congestion level recorded in the data packet from any multiplexing path falls below the *path teardown threshold*, then the initiator tears down the multiplexing path.

3.3. Packet Multiplexing

Incoming traffic are basically multiplexed in a round-robin fashion by the traffic dispatcher. However, if a neighbor of the traffic dispatcher on one of multiplexing paths

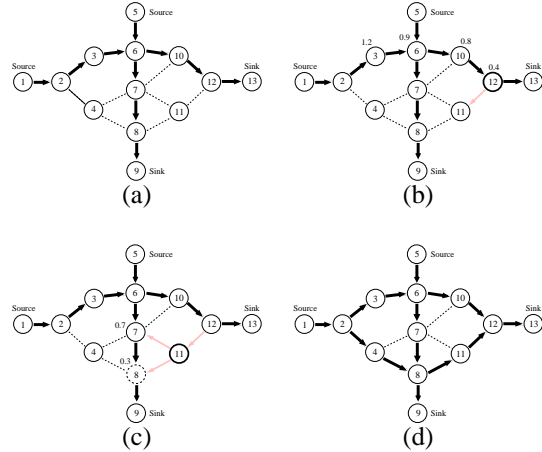


Figure 2. Procedure of selecting a multiplexing path

suffers bad channel condition possibly by its own neighbor, the incoming packet can be more preferably forwarded to the neighbors with a better channel condition, preventing the head-of-line packet from blocking the transmission of enqueued packets. However, this requires a MAC-layer assistance such as the number of retransmission or backoff window size to infer the channel quality. In addition, the bias for better channel condition should be limited since the good channel to a neighbor does not imply high channel capacity along the downstream nodes after the neighbor.

3.4. Multiplexing Path De-construction

The initiator keeps scanning the congestion level of the multiplexing path by inspecting the per-flow congestion level recorded in the data packets from the multiplexing path. If that value from any multiplexing path falls below the *path teardown threshold*, then the initiator starts to deconstruct the multiplexing path by sending a request message to the neighbor on the multiplexing path. The nodes on the multiplexing path already have a pointer to their preceding node, which was established during the path selection. The node on the multiplexing path simply forwards the request message to its preceding node while forwarding data packets. When the traffic dispatcher receives this request message, it stops forwarding data packets onto the multiplexing path and send a reply message onto the multiplexing path. When the nodes on the multiplexing path receive the reply message, they forward the reply message and stop forwarding data packets. The path de-construction is finished when the initiator receives the reply message.

6	7	8
3	4	5
0	1	2

Figure 3. A 530m by 530m sensor field with 9 grids

4. Simulation Results

4.1. Performance Metrics

In this paper, we consider the following two important metrics to evaluate our proposed schemes: *delivery quality* and *energy efficiency*.

- *Delivery Quality*: As far as applications are concerned, delivery quality is the most important metric. In this paper, we use the event delivery ratio to measure the delivery quality. Specifically, we will count the total number of events delivered by all the sources, and the total number of events received by the sinks. The difference between these two signifies the number of events that are dropped within the network.
- *Energy Efficiency*: We evaluate the energy efficiency of our protocol using the total amount of energy spent in sending all the packets (including both data and control packets).

4.2. Simulation Environments

802.11 DCF with RTS/CTS is used for our MAC protocol. Up to 7 retransmissions are allowed for a collided packet. The packet size is 52 bytes. The buffer capacity of a node is 15 packets. The raw channel bandwidth is 2 Mbps. Directed diffusion [5] is used as an underlying routing protocol. To make a dense network with a strong connectivity, we randomly deployed 225 nodes with a radio range of 50m in the sensor field shown in Figure 3. In each instance of simulation, two sinks from different grids except the center grid (grid 4) are randomly picked and for each sink, three corresponding sources are also randomly chosen from three different grids except the center grid in the sensor field shown in Figure 3. Each instance of simulation runs for 70 seconds. The three sources send one packet per second to their sink in a dormant state and take turns to start reporting high rate samples during one of 10-20, 30-40, or 50-60 time period, which we consider crisis states. The other three sources follow the same data traffic pattern, so that during each of 3 crisis states (10-20, 30-40, and 50-60 time periods) the flows from different sources can create a hotspot.

4.3. Delivery Quality

The required event accuracy level is application-dependent. Here, we simply assume the delivery ratio of samples to the sink determines the accuracy level of an event, which indicates the accuracy level increases linearly with the delivery ratio.

Figure 4(a) shows the number of delivered packets observed by the sinks. To quantify the effect of congestion, we placed the sources and sinks so that no congestion occurs, which we define as the *ideal* case. In this ideal case, the packet delivery rate is bounded only by the channel capacity if each node has enough queue capacity. It has been shown that the observed capacity of the IEEE 802.11 MAC protocol in a chain of nodes is as high as $\frac{1}{7}$ of a single cell capacity [8]. To see our results in the ideal case in Figure 4(a) produce the throughput suggested by [8], we compare the throughput in the ideal case with the calculated throughput according to [8]. This can be a sanity check for our node deployment. Since RTS, CTS, DATA, and ACK packets are 44, 38, 52, and 38 bytes, respectively and the raw channel capacity is 2 Mbps, the single cell capacity of a node in our sensor field is $\frac{52}{44+38+52+38} \times 2 \text{ Mbps} \approx 600 \text{ Kbps}$. Therefore, 85 Kbps ($\approx \frac{600 \text{ Kbps}}{7}$) is the observed string capacity according to [8]. The highest throughput in the ideal case is obtained when the data rate is 250 packets/sec (around 100 Kbps). However, 10% of packets are dropped as shown in Figure 4 (b), which makes the throughput around 90 Kbps in a crisis state. This is similar to what we calculated (85 Kbps).

In each instance of simulation, we limit the number of multiplexing paths per each hotspot to 1 and 2, which are drawn on “1 alternative path” and “2 alternative paths” curves, respectively. Compared with the case without resource control, a single multiplexing path enhances the accuracy level by 33.3% while two multiplexing paths provides 50% of accuracy enhancement at 300 packets/sec in Figure 4(a). Figure 4(b) shows the packet drop rate measured in the whole network. The packet delivery rate is defined as $(1 - \text{packet_drop_rate})$. If the application cannot tolerate 10% of packet drops during congestion, then without resource control, data rate should be reduced at 150 packets/sec as shown in (b). However, this traffic control will also reduce the delivered packets shown in (a), lowering the accuracy level. This clearly shows the traffic control strategy sometimes cannot be deployed if we have available resource.

4.4. Energy Efficiency

The increased energy consumption by deploying more nodes on the routing path should not exceed the delivery quality gain.

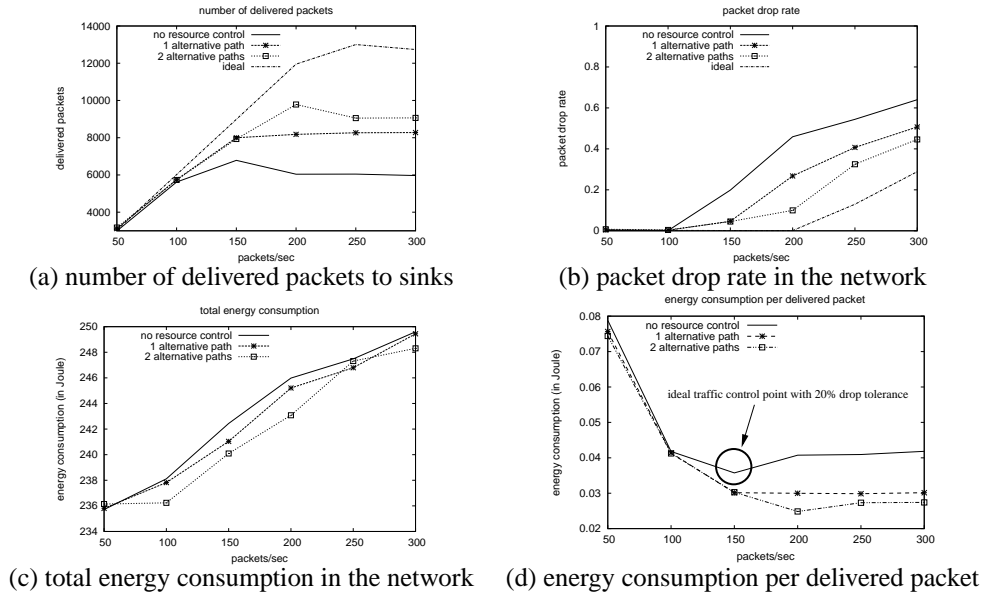


Figure 4. Delivery quality and energy efficiency of resource control

Figure 4(c) shows the total energy consumed in the network. Contrary to intuition, our resource control consumes less energy compared to the case without resource control. This is because a lot of energy is wasted by collisions and subsequent retransmissions for the collided packets.

Figure 4 (d) shows how much energy is spent to route a single packet to the sink. We call it *energy-per-packet*. It shows having even a single additional path significantly reduces the energy-per-packet. This shows the additional energy consumed by our scheme offsets the energy wasted due to packet drops. This result shows resource control scheme does not incur extra energy consumption as long as it reduces the packet drop rate.

Now, let's compare our resource control scheme with an ideal traffic control scheme. If the application tolerates 20% of packet drops, then an ideal traffic control scheme will stop increasing incoming traffic volume at 150 packets/sec as already shown in Figure 4(b). At this point, the incoming traffic fully takes advantage of channel capacity. The energy-per-packet at this point is around 0.036 J as shown in Figure 4(d) while our resource control scheme spends 0.03 J to deliver a packet. This clearly shows the energy consumed to deliver a single packet to the sink even under the ideal traffic control scheme exceeds that of our resource control scheme.

This result also shows that successfully delivering one packet during low traffic consumes significantly more energy than during high traffic volume, the reason being that it spends most of energy idle-listening. It indicates not only some energy-conserving scheme should be used when measured congestion level is low, but also our adaptive resource

control scheme, which comes into play only during congestion, is *energy-friendly*.

5. Concluding Remarks and Future Direction

In this paper, we proposed a resource control scheme that adaptively adjusts the resource provisioning by creating multiple paths when congestion is detected. To deal with ongoing congestion, we increase more resources to meet the demand of resource. Unlike the traffic control that tries to reduce the incoming traffic into the hotspot, our resource control scheme maintains a constant level of throughput that is critical in a crisis state. Our simulation results show when congestion is transient, increasing resources by creating multiple paths around the hotspot effectively increase not only the delivered packets (accuracy level), but also saves a lot of energy by avoiding collisions and retransmissions.

Creating multiple data paths is one way of increasing resource. Adjusting radio range or changing radio modulation scheme can be another option for resource control. In addition, for highly transient congestion more short-term resource control may be needed.

We'll also investigate the hybrid congestion control scheme that triggers either traffic control or resource control based on some criteria such as resource availability and accuracy level required by applications.

References

- [1] A. Cerpa and D. Estrin. ASCENT: Adaptive Self-Configuring Sensor Networks Topologies. In *Proceedings of IEEE INFOCOM'02*, June 2002.
- [2] S. De, C. Qiao, and H. Wu. Meshed multipath routing with selective forwarding: an efficient strategy in wireless sensor networks. In *Elsevier Computer Networks, Vol. 43, pp. 481-497*, 2003.
- [3] S. Dulman, T. Nieberg, J. Wu, and P. Havinga. Trade-Off between Traffic Overhead and Reliability in Multipath Routing for Wireless Sensor Networks. In *IEEE Wireless Communications and Networking Conference*, March 2003.
- [4] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks. In *ACM Mobile Computing and Communications Review, Volume 5, Issue 4, pp. 11-25*, 2001.
- [5] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networks (MobiCOM)*, August 2000.
- [6] S. J. Lee and M. Gerla. Dynamic Load-Aware Routing in Ad hoc Networks. In *Proceedings of IEEE International Conference on Communications (ICC)*, June 2001.
- [7] S. J. Lee and M. Gerla. Split Multipath Routing with Maximally Disjoint Paths in Ad hoc Networks. In *Proceedings of IEEE International Conference on Communications (ICC)*, June 2001.
- [8] J. Li, C. Blake, D. S. J. D. Couto, H. I. Lee, and R. Morris. Capacity of Ad Hoc Wireless Networks. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2001)*, July 2001.
- [9] M. K. Marina and S. R. Das. On-demand Multipath Distance Vector Routing in Ad Hoc Networks. In *Proceedings of International Conference for Network Protocols (ICNP)*, November 2001.
- [10] A. Nasipuri, R. Castaneda, and S. R. Das. Performance of Multipath Routing for On-Demand Protocols in Mobile Ad Hoc Networks. In *ACM/Kluwer Mobile Networks and Applications (MONET), vol.6, no.4, pp. 339-349*, 2001.
- [11] M. R. Pearlman, Z. J. Haas, P. Sholander, and S. S. Tabrizi. On the Impact of Alternate Path Routing for Load Balancing in Mobile Ad Hoc Networks. In *Proceedings of ACM MobiHoc*, August 2000.
- [12] Y. Sankarasubramaniam, O. Akan, and I. Akyildiz. ESRT: Event-to-Sink Reliable Transport in Wireless Sensor Networks. In *proceedings of the ACM MobiHoc Conference*, 2003.
- [13] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman. Infrastructure tradeoffs for sensor networks. *ACM Workshop on Wireless Sensor Networks and Applications (WSNA)*, Sep. 2002.
- [14] Y. Tseng, C. Hsu, and T. Hsieh. Power-Saving Protocols for IEEE 802.11-Based Multi-Hop Ad Hoc Networks. In *Proceedings of IEEE INFOCOM'02*, June 2002.
- [15] A. Valera, W. K. G. Seah, and S. Rao. Cooperative Packet Caching and Shortest Multipath Routing in Mobile Ad hoc Networks. In *Proceedings of IEEE INFOCOM*, March 2003.
- [16] C. Wan, S. Eisenman, and A. Campbell. CODA: Congestion Detection and Avoidance in Sensor Networks. In *proceedings of the ACM SenSys 2003*, 2003.
- [17] K. Wu and J. Harms. Performance Study of a Multipath Routing for Wireless Mobile Ad Hoc Networks. In *Proceedings of IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, February 2001.
- [18] T. Yan, T. He, and J. A. Stankovic. Differentiated Surveillance Service for Sensor Networks. In *Proceedings of the ACM SenSys 2003*, 2003.
- [19] F. Ye, G. Zhong, S. Lu, and L. Zhang. PEAS: A Robust Energy Conserving Protocol for Long-lived Sensor Networks. In *Proceedings of the 23rd International Conference on Distributed Computing Systems*, May 2003.
- [20] W. Ye, J. Heidemann, and D. Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proceedings of IEEE INFOCOM'02*, June 2002.
- [21] Z. Ye, S. V. Krishnamurthy, and S. K. Tripathi. A Framework for Reliable Routing in Mobile Ad Hoc Networks. In *Proceedings of IEEE INFOCOM*, March 2003.
- [22] R. Zheng, J. C. Hou, and L. Sha. Asynchronous Wakeup for Ad Hoc Networks: Theory and Protocol Design. In *proceedings of the ACM MobiHoc Conference*, 2003.
- [23] R. Zheng and R. Kravets. On-demand Power Management for Ad Hoc Networks. In *Proceedings of IEEE INFOCOM'03*, March 2003.