Cluster-based Congestion Control for Supporting Multiple Classes of Traffic in Sensor Networks

Kyriakos Karenos, Vana Kalogeraki and Srikanth V. Krishnamurthy
Department of Computer Science and Engineering
University of California, Riverside
Email:{kkarenos,vana,krish}@cs.ucr.edu

Abstract

In wireless sensor networks, multiple flows from data collecting sensors to an aggregating sink could traverse paths that are largely interference coupled. These interference effects manifest themselves as congestion, and cause high packet loss and arbitrary packet delays. This is particularly problematic in event-based sensor networks where some flows are of greater importance than others and require fidelity in terms of higher packet delivery and timeliness. In this paper we present COMUT (COngestion control for MUlti-class Traffic), a distributed cluster-based mechanism for supporting multiple classes of traffic in sensor networks. COMUT is based on the self-organization of the network into clusters *each of which autonomously and* proactively *monitors congestion within its localized scope. The clusters then exchange appropriate information to facilitate system wide rate control. Our simulation results demonstrate that our techniques are highly effective in dealing with multiple, randomly initiated flows.*

1. Introduction

In this paper we present a scalable and distributed framework for eliminating congestion and supporting multiple classes of flows in *event-based* sensor networks. In contrast to monitoring applications, wherein sensors are deployed to report periodic data, in *event-based* sensor networks, reports are produced only upon the observation of specific events that satisfy certain pre-specified conditions; a typical example might be the increase in the observed temperature beyond a preset threshold.

We consider sensor networks that consist of a relatively large number of cheap, disposable sensors which report to only a small number of aggregating sinks. Collisions of packets from simultaneous, interference-coupled flows create congested hotspots which, in turn, cause flows to experience delays and packet drops. The problem becomes more critical in applications such as disaster recovery missions, where packets from some flows are likely to be of greater importance than others. Maintaining a high delivery ratio for the more important flows is critical in these networks. Our work targets these scenarios and has two specific but inter-related constituent objectives: (i) provision of distributed mechanisms for congestion control and, (ii) management of flows from multiple classes, i.e., of higher versus lower importance.

Traditional congestion control approaches utilize end-to-end or hop-by-hop (or combinatory) techniques [4, 15] but consider only a single class of packets. End-to-end techniques [13] require the sink to regulate the sensors' sending rate. However, because traffic volume is higher in the proximity of the sink, the regulatory updates sent by the sink may be throttled at the source. On the other hand, hop-by-hop, backpressure techniques [15] are reactive in nature and might not create responsiveness in a timely fashion.

Previously proposed service differentiation techniques have not considered congestion or its effects [11, 18]. Admission control techniques proposed for wireless, ad hoc networks [11, 19], consider the network load and thus regulate congestion indirectly. However, these methods are likely to be computation and overhead intensive in the presence of multiple classes of flows and will, hence, be unsuitable for sensor networks.

In this paper we present COMUT (COngestion control for MUlti-class Traffic), a framework that provides scalable and distributed cluster-based mechanisms for supporting multiple classes of traffic in sensor networks. In the techniques previously discussed, congestion is estimated and action is taken on a *per-node* basis. The distinguishing characteristic of our approach is that COMUT is based on the self-organization of the network into *clusters* each of which autonomously and *proactively* monitors congestion within its localized scope. To accomplish this, *sentinel* roles are assigned to sensors to proactively monitor network statistics and infer the *collective* level of congestion. Regulation of sensor rates (*per-cluster*) and coordination between clus-

ter nodes is achieved by exchanging only small volumes of control information between the sentinel sensors along flow paths. Sensor clustering is beneficial in that a group of sensors can capture the behavioral interactions between flows. The sensors in a cluster adjust their rates as per the relative level of importance of the events to be reported and the congestion state en route the sink, thus improving the timeliness of data delivery for high importance flows and the efficiency with which the available bandwidth is shared between the flows. This process improves the timeliness of data delivery seen by flows of high importance while improving the efficiency with which the available bandwidth is shared between flows.

We summarize our contributions below: We propose a framework for congestion and rate control in highly dynamic and unpredictable event based sensor systems wherein multiple classes of flows are to be supported. Our framework consists of the following components: **(i)** A distributed and scalable mechanism that facilitates the clustering of sensors and allows for the adjustment of the sending rate per cluster. **(ii)** A decentralized methodology for intra- and inter-cluster, *per-path* estimation of traffic intensity. The estimation is *proactive* and helps anticipate fluctuations in bursty traffic patterns. Through an extensive set of simulations we demonstrate that our mechanisms are highly effective in supporting flows of multiple priorities and in achieving high delivery ratio of important packets.

2. Design Objectives

In this section, we motivate our approach and highlight the factors that influence our design decisions. Throughout this paper we assume that *the scenario of interest is akin a disaster rescue mission.*

Coping with the presence of interference coupled paths: In event-based sensor environments such as the one under consideration, the number of scuttered sensors is relatively large, while the number of aggregating sinks is likely to be small. This leads to a funneling effect wherein data flows from different sensor-detected events converge when they reach the proximity of the sink. The flows are likely to follow paths that are largely interference-coupled while sources are unaware of the existence of other flows in the network. The problems are further exacerbated in the presence of flows belonging to multiple classes, since high importance events may get lost during congestion. Furthermore, it may be impossible to bypass congested areas that are close to the sink. In the presence of such interfering flows, our objective is to give preference to and maximize the throughput of higher importance flows.

The need for proactive rate control: In disaster recovery missions, events are likely to be detected at random times and at random places in the sensor network, thereby dynamically changing the interference patterns among flows. This characteristic hinders our ability to predict system behavior and effectively identify points of congestion. In addition, the presence of multiple classes of traffic makes the problem even more difficult. One might use reactive techniques to coerce the sources to reduce their rates (especially sources that inject traffic of lower importance) upon congestion. However, in such cases congestion has already occurred, thus reactivity does not avoid critical packet loss. Our goal is to *proactively* monitor the network and identify or predict the onset of congestion in a cluster. A lightweight control scheme is also required for the information to be quickly exchanged between clusters to notify the sources that route packets through the congested area to reduce their sending rates. In addition, in order to ensure that events related to congestion are less likely to occur, it is important for source nodes to be conservative in terms of their transmission rates when they begin injecting traffic (especially those sources that generate low priority data). In summary, proactive rate control anticipates unpredictable injection of flows and avoids dropping critical packets upon congestion.

The need for Cluster-based Traffic Intensity Estimation: Given that we need an efficient, proactive rate control scheme, our objective is to define a simple yet effective congestion detection mechanism which will consider the existence of multiple types of flows. *Towards the detection of congestion we then need a methodology for performing traffic intensity estimation*. Such an estimation technique must be simple, decentralized, lightweight and should be implemented efficiently within the sensors' *localized* scope. In order to reduce the traffic load without, however, having to discard packets, it will be necessary to adjust the rate of the sources themselves. Therefore, the traffic intensity estimates must be propagated to the event sources, which may not be known a-priori. One possibility is that this intensity metric be communicated to all sensors in the network in control packets. However, this would be energy costly and unnecessarily increase the network traffic, thus, further contributing to congestion. To implement a scalable solution, in our approach sensors are grouped in clusters that collaborate by exchanging information on observed, active flows and their corresponding importance. Grouping the sensors into clusters assists in capturing traffic from multiple flows, thus and produces a more accurate representation of flow interactions.

3. Methodology

As stated, our mechanisms are employed on per cluster bases. We first describe, in this section, how clusters are formed. Once formed, the clusters monitor the network to estimate the level of localized congestion and take appro-

priate actions based on the observations from the monitoring process.

Cluster Formation: In COMUT, sensors *self-organize* into clusters. Each cluster is governed by an appointed sentinel. Sensors make traffic estimates and send these estimates to the sentinel via a *single-hop* broadcast; the estimates are then processed by the sentinel as will be discussed later.

Sentinel Election: During the *election phase* initially, each sensor node sets a time-out at a pre-defined, randomly chosen, later time t_s. If within this period, it receives a *sentinel announcement message* from a neighboring node, it chooses the sender to be its sentinel and joins that sentinel's cluster[1]. If, after the lapse of t_s, no message is received, the sensor will become a sentinel itself with a probability P_n. With a probability $(1 - P_n)$, it simply sets a new time-out, to expire after a randomly chosen period of t_s. P_n is a function of n, where n is the count of the number of instances when a sensor node, after timing out, chose *not* to elect itself a sentinel. P_0, the initial probability, is chosen to be a small random value. If a sensor node does not succeed in becoming a sentinel and it has not joined a cluster, it increases P_n to P_{n+1}, at the end of the next time-out, as per the following formula:

$$P_{n+1} = (1 - P_n)(1 - e^{-\alpha n}) + P_n$$

where α is a parameter that determines the effective degree of increase of P_n with n. Note here that the increase in P_n becomes more significant as n increases. Thus, within a relatively small number of iterative steps, each node either becomes a sentinel itself or a member of a cluster headed by a neighbor sentinel[2].

Routing Functions: In COMUT, a routing protocol that allows sensors to discover and store information with regard to their neighborhoods is needed. The information aids the formation of clusters. This functionality is supported by most sensor or ad-hoc routing protocols [3]. The Zone Routing Protocol (ZRP) [6] is a protocol that is especially attractive for facilitating the formation of clusters and hence is chosen as the routing protocol in this work. ZRP is a routing protocol that combines both proactive and reactive routing approaches. Proactive information is maintained within neighborhoods, referred to as *zones*, that are constructed around each node. The proactive part of the protocol allows for the quick forwarding of *traffic control messages* to the sentinels when they are located within the zone radius. The routing framework thus provides an inherent structure that can be exploited. The overhead with ZRP increases with zone radius; however, with an increased radius, the sentinels can exchange messages with a lower latency. Furthermore, route queries are not needed at each instance of a control message exchange.

Traffic Intensity Estimation: Once the clusters are formed, the objective is to determine the level of local congestion within each cluster. This estimate will then be fed back to the sources of the data flows so that they can appropriately control their sending rates. To determine the level of congestion in the sensor network it is important to estimate the *traffic intensity* both within and across multiple clusters. The traffic intensity is significantly affected by the number of *new* incoming and existing flows, the density of nodes in the network and the limitations in terms of computation abilities of sensor nodes. Thus, this estimation is very hard to achieve in a practical setting. We use a queuing network model, wherein each sensor is modeled as a queue. The queues are then interconnected within a cluster to form a network. We assume that the resulting network is a BCMP network of queues [7]. This enables us to represent the state of the network in a product form which would be the case if the queues were each examined independently and in isolation. This model although approximate, lends simplicity and tractability and is adequate for our purposes which is to acquire *macroscopic* network statistics. The simplicity of our approach entails minor computational and communication overheads making it feasible for use in sensor networks. Furthermore, the model also provides estimates that are within acceptable accuracy levels as will be demonstrated later when we discuss our simulation results (Sec. 4). Using the above model, we compute the traffic intensity as follows:

Let the value ρ_i represent the offered load at the queue of sensor i defined as $\rho_i = \frac{\lambda}{\mu_i}$ where λ is the arrival rate of the packets from the flow of interest and μ_i is the service rate at sensor i. The distribution of the queue size in terms of the probability that there are k packets in the queue, $P(k)$, at the sensor node is then [7]:

$$P(k) = (1 - \rho_i)\rho_i^k$$

For *N* distinct queues, the joint distribution is the product:

$$P(k_1, k_2, ..., k_N) = \prod_{i=1}^{N} (1 - \rho_i){\rho_i}^{k_i}$$

We thus, specify the *traffic intensity* estimate (also called the γ estimate) in a cluster, γ, to be the probability that at least one of the sensors in the cluster has a non-empty queue:

$$\gamma = 1 - P(0, 0, ..., 0) = 1 - \prod_{i=1}^{N} 1 - \rho_i$$

Given the above definitions, each sensor i estimates its local load ρ_i within a specific time interval, and reports it to its sentinel via a local broadcast. Note here that these estimates are made based on real time observations of packet

1 With CSMA, a sensor cannot send a message when it senses a busy channel. Upon time-out expiry, if a sensor receives a sentinel announcement, it will join the sender's cluster and suppress it's own announcement.

2 Simulation experiments indicate that, even for networks with as many as 200 nodes, with a value of $\alpha = 0.2$, a convergence is reached in at most 10 iterative steps.

arrivals and departures. Upon collecting the values ρ_i, from all of the sensors, where, $1 \leq i \leq N$, in the cluster, the sentinel calculates the collective γ estimate for the cluster. Similarly, an aggregate estimate of the traffic intensity on an entire path from a set of sources to a sink is calculated via the exchange of the γ values between the sentinels on the path. This value is used as an indicator of the congestion level. If a single sensor within a particular cluster is overloaded, the value of γ will be high and the cluster will be marked as congested. Consequently, a congested cluster will, in tern, identify congested flows passing through that cluster.

We wish to point out that contention for the wireless channel at each sending and receiving sensor node affects the traffic intensity; if contention is high then, a packet that reaches the *head* of a queue takes a longer time prior to receiving service.

We now need to find a threshold value for γ beyond which the cluster is considered to be congested. The average number of packets in the system can be theoretically calculated with increasing load ρ [7]. The analysis shows that when ρ is below the value of 0.6, the average number of packets in the system increases relatively slowly, while for values greater than 0.6 the system saturates rapidly. For clusters of sizes up to 10, assuming each sensor's estimated load ρ_i to be between 0.5 and 0.6, the value of the threshold γ_{thres} is approximately 0.95. In our simulation experiments we observed that these values provided a high degree of accuracy in estimating the traffic intensity for different sizes of clusters. This is because, in cases of congestion, the deviation in γ with a varying cluster size is small. We reiterate that this value is an estimate: it indicates the probability of finding at least a single packet in a cluster. Our goal is to keep γ below 0.95. For higher values of γ, the associated path that traverses the cluster is marked as congested.

Rate Regulation: Once the traffic intensity along a path is estimated, our objective is to regulate the rate at which source sensors send their data toward the sink. The process of exchanging the intensity information and using it to regulate the traffic in the network, is deliberated below.

Initialization: Consider the sensor network at setup time. Since flows are yet to be initiated, sentinels do not have any knowledge of where congestion is going to occur and where to send updates. Flooding updates to all sentinels within a preset neighborhood wastes energy without guaranteing that congestion will be prevented. For a prospective flow, we define the *upstream* path to be the path created by the routing protocol from a source cluster to the sink. The path is identified simply by the clusters that its packets traverse[3]. Sentinels, then, need to only send updates to upstream clusters, i.e., toward the source cluster. In order to identify the set of such upstream sentinels, in this phase, each sentinel sends a small control packet to the sink; the packet carries the cluster ID of the sending sentinel. As packets flow towards the sink, each sentinel overhears the transmissions of packets that pass through its cluster and thereby identifies its upstream clusters.

One important question is that prospective sources must have an estimate of the time interval to wait for an update from the sentinel in their cluster, indicating the congestion level along the path of the flow so as to regulate their sending rates. *This information is needed by the sources to decide upon the volume of additional traffic that can be injected into the network.* We use an empirical method to estimate this time and we refer to it as the *Rate Regulation Epoch (RRE)*. RRE is computed at the sink by calculating the total delay experienced by a packet[4]; the information is then communicated back to the particular sentinel that originated the packet. Although RRE is measured under specific network conditions, by using weighted moving averages, the sink adjusts the estimate during network operations, and periodically transmits the new estimates.

Intra- and Inter-Cluster Communication: Sensors within a cluster periodically estimate their current load ρ_i. Within an estimation interval t_e each sensor (i) measures the arrival rate by counting of the locally incoming packets and (ii) estimates the service rate from the total time taken to complete the servicing process for all packets arriving in t_e. Transmission of the estimation is *not* required to be performed at the end of each estimation interval. As discussed earlier, in order to save energy, a sensor transmits its computed value to the sentinel only if it exceeds a present threshold (0.6). Lower values are ignored since they do not impact whether or not there is congestion on the path. The sentinel will periodically forward its locally computed traffic intensity value (based on the information from the sensors in its cluster) and the level of the highest importance flow observed in its cluster, to other sentinels on the upstream path followed by each flow passing through the cluster. This is an indication of the level of congestion of the specific path to the upstream sentinels (with respect to the sentinel that sends the update) on that path. Again, in order to save energy, a sentinel will trigger the aforementioned periodic forwarding of the intensity information only if the measured intensity exceeds a preset threshold discussed later (see also Table 1).

Although our techniques are designed with the objective to reduce the number of control messages and, thus, energy consumption, periodic broadcasting in sensor networks clearly incurs energy expenditure. However, broadcasting is inherently deployed in many existing sensor systems (*e.g.*,

3 A cluster is effectively identifi ed by its sentinel's ID. By *cluster update communications*, we essentially refer to communications between sentinels.

4 While computing this one-way delay, we assume sensor synchronization, achieved with techniques such as in [5].

periodic neighborhood beaconing in motes' routing[17], estimating per-hop delays in SPEED [8]). In our experiments, we show that the energy consumed due to communication overhead is in fact compensated by saving energy that would otherwise be wasted due to transmissions of packets that are eventually dropped due to congestion.

Rate Self-Regulation at the Sources: Finally, in response to the messages received, the rate at which packets are generated at a source cluster is to be regulated. An adaptive regulation mechanism is key to controlling the congestion level in the network. We propose a adaptive scheme that follows a regulation policy that is similar to the popular Additive Increase, Multiplicative Decrease (AIMD) policy with TCP [10]. In our technique, however, the rate is dropped to a minimum $minrate$ for low importance packets (instead of multiplicative decrease), if packets of higher importance exist along the path followed by the source's flow *or* upon estimating congestion. The minimum rate would be a preset value that can be measured experimentally or via simulations and can be set before the sensors are deployed.

We deliberate on the regulation process in more detail. We define $d_p > 0$ to be a value associated with each level of importance p. This value determines the number of RRE time intervals that a sensor should wait before it increases its rate (as discussed earlier). For higher importance events the d_p value will be set to be smaller than the one set for lower importance flows. This "slow start"-like scheme aims at waiting for the network to respond to the currently injected traffic before introducing additional load. The increase is additive and repeats for *m* RRE's after which the rate is dropped again to the minimum. However, if during the rate increase either the default maximum rate $maxrate$ is reached or the cluster traffic intensity exceeds γ_{thres}, then the rate is again dropped to the minimum. To summarize, our scheme anticipates the injection of dynamic flows in the network and proactively regulates the rate while waiting for congestion feedback.

4. Evaluation

Simulation Settings: We have evaluated the performance of COMUT running simulations on the Network Simulator (NS) tool [2]. We generated random network topologies in an square area of 100*m* x 100*m*. The number of sensors was varied between 60 and 140 in steps of 20. One of the sensors was selected at random to be the sink. The simple CSMA based MAC protocol, with an exponential backoff policy was assumed; this has been generally used in prior work on sensor networks [13][15][14] that do not specifically address the MAC layer. The sensor nodes are set up to simulate the characteristics of MICA motes [1]. They are homogeneous and have a transmission range of 25*m*. Data packets are 30 bytes (the standard size). The memory in a mote is 4KB; thus, the queue size is restricted to accommodate at most 65 packets. We assume the presence of robust physical layer techniques to cope with bit errors; hence, packet drops are attributed only to queuing related drops and to collisions. We simulated a scenario where multiple consecutively occurring events are sensed at random points on the sensor field (an *event burst*). The events are assigned either a *high* or a *low* importance level and event reports are initiated in time steps of *500ms*. Additional parameters used are listed in Table 1.

Number of Sensors	60..140
Area	100m x 100m
Sensor Transmission Range	25m
Queue Size	65 packets
Data Packet Size	30 bytes
Minimum Rate	1 kbps ($\approx$ 2.5 packets/sec)
ZRP radius	3 hops
γ_{thres}	0.95
Update trigger thresholds	Sentinel: 0.8, Sensor: 0.6
Event Burst time Simulated	20sec

Table 1. Parameters Used in Simulation

In order to demonstrate the interference coupling between high and low importance flows, we first initiate three low importance events followed by three high importance events and then, another two low importance events, i.e. a total of eight flows. We compared our techniques (with-COMUT experiments) with scenarios where there is no congestion control (no-COMUT experiments). In both the no-COMUT and with-COMUT experiments, sensors that detect an event send packets at constant bit rate (the rate at which packets are sent is a parameter that we vary) for *10sec*. The results presented are averaged over 3 runs with random topologies, for each chosen network density.

Evaluation of the γ Estimator: In the first set of experiments we evaluated the accuracy with which the traffic intensity estimator (γ estimator) can estimate the traffic congestion in the sensor network. For this, we set up a grid network containing 196 nodes with a single sink placed in the middle of one side, as shown in Fig. 1. The purpose of setting up a grid is to be able to easily identify the clusters formed and the flow paths produced. We introduce two flows and ensure that they pass through a common cluster. We statically identify the clusters via which the flows pass. We plot the value of the *collective* γ estimator at each of these clusters as the rate increases and as measured by the cluster sentinel within an observation period of 3*sec*.

We present, in Fig.2, the observed γ values on the path traversed by one of the two flows versus the rate at which packets are sent. The results for the second flow are similar. The estimation is done over a period of 3 seconds; within this time, we observe transient behaviors that help us under-

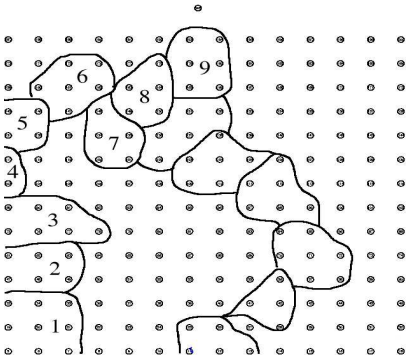

Figure 1. Network Flow Clusters.

stand our mechanisms. The clusters where the flows collide are 8 and 9. Congestion is experienced at these clusters even if the individual rates of the flows are low ($\approx$ 5 Kbps) and *early* during the 3 second observation period; thus, packets are dropped. However, due to delays resulting from congestion, packets are filling up the queues at sensors on the path prior to reaching Clusters 8 and 9. Therefore, at later times, congestion is evident at Cluster 5. Note that these observations are within individual clusters. Since, with our framework, we perform a collective estimate across multiple clusters, a single congested cluster is sufficient to indicate a state of congestion for the entire flow along the path.

Note also, that there is a significant change in the γ estimate when we move from a low to a high load. γ increases rapidly at high loads (above 0.8), while it remains low (below 0.1) for rates that produce low levels of congestion. This suggests that we are able to save energy since we do not need to send updates during stages of zero or low congestion levels. The periodic updates are to be initiated only if γ exceeds a specific threshold.

System Operations: Our next experiment demonstrates the transient behavior of the system when our framework is in place. Fig.3 shows the sending rate at the sources as a function of time; flows are introduced in a random network of 60 nodes. The *RRE* is set to *0.1sec* and the d_p values (Sec. 3) are 1 for high and 2 for low importance packet flows. Initially, a low importance flow ("Low1") is injected at time 0. Fig.3 shows that the rate at which the sources send packets of this flow increase to begin with. However, after a high importance flow ("High") is started 2 seconds into the simulation, the source of "Low1" was informed of the advent of the new flow, and consequently, it drops its rate to a prespecified minimum. Notice that "High" flow increases its rate quickly while "Low1" flow keep its rate low. When "High" reaches a rate of *7kbps* congestion is detected and as a result, the rate for "High" is dropped to the minimum. Note that "Low1" still keeps "silent" and allows for "High" to increase its rate again. At time *6.6sec* we introduce a second low importance flow "Low2", this time at the point of congestion (where the monitored γ estimate is high). However our methods are still effective and "Low2" holds its rate to the specified minimum. This demonstrates that regardless of the position of the *origin* of a flow, our framework succeeds in controlling the rates as appropriate.

Received Packets: In Fig.4 we plot the number of packets received per flow over the entire simulation time as a function of increasing *maxrate* value. We show that in both sparse (60 nodes) and dense (140 nodes) networks we achieve a higher delivery of packets for the high importance flows as compared to that for the lower importance flows. We also note that beyond a certain value of *maxrate*, the delivery rate seems to stabilize to a constant value. With COMUT, at a certain rate, congestion begins and flows are not allowed to increase their rates beyond this rate.

Delivery Ratio: Another important property of COMUT is showcased in Fig.5. The figure plots the observed delivery ratio with and without COMUT. The rate without COMUT reflects the constant rate at which the sources send. The results are for dense networks and we have obtained similar results for sparse networks. Note that the delivery ratio for COMUT remains above 0.8 in all cases for both high and low importance flows. Note also, that the number of packets for higher importance flows is greater than that for the lower importance ones as shown in Fig.5. Without our congestion control techniques, we do notice a high percentage of dropped packets. For each packet delivered a large number of packets are dropped which indicates significant wastage of energy.

Average Per-hop Delay: We also plot the average per-hop delay for all flows with and without COMUT. We observe, in Fig.6, that in both sparse and dense networks, with COMUT the delay is significantly lower. The reason is that since congestion is reduced, high queuing delays are avoided.

Energy Dissipation: In the last set of experiments we demonstrate the energy savings due to COMUT. For each packet dropped when COMUT is not employed, we measure the wasteful transmissions by counting the number $T_{Dropped}$ of MAC packet transmissions and retransmissions due to forwarding over multiple hops times the size of the data packet S_d (in bits). This is the total number of wasted bit transmissions: $D = T_{Dropped} \cdot S_d$. Note that this product essentially reflects the wasted energy due to dropped packets. With COMUT case we must also consider the (re)transmissions of control packets $T_{Control}$ of size S_c. Thus, the expended energy due to wasted transmissions and control traffic is calculated as: $D_{COMUT} = T_{Dropped_{COMUT}} \cdot S_d + T_{Control} \cdot S_c$. We represent the energy savings with COMUT using the following ratio: $R = \frac{D - D_{COMUT}}{D_{COMUT}}$. We plot *R* in Fig.7 with increasing sending

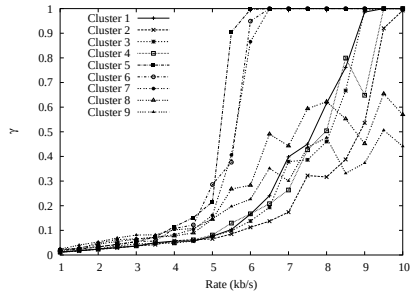

Figure 2. γ Estimator for all Clusters vs Rate.

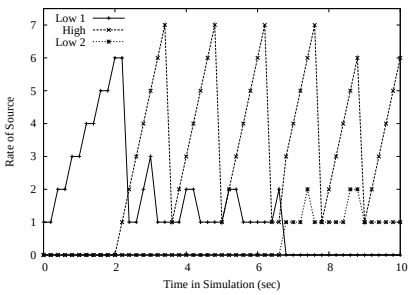

Figure 3. Time trace for the Sending Rate of Sources.

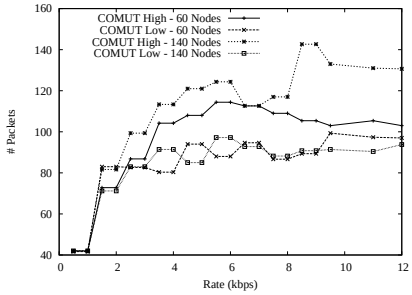

Figure 4. Received Packets vs Rate (High & Low Density).

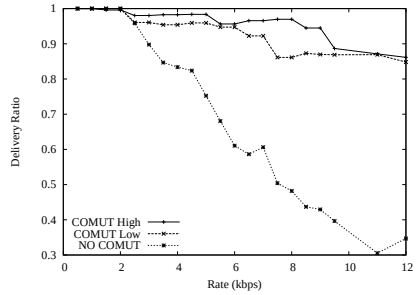

Figure 5. Delivery Ratio vs Rate (High Density).

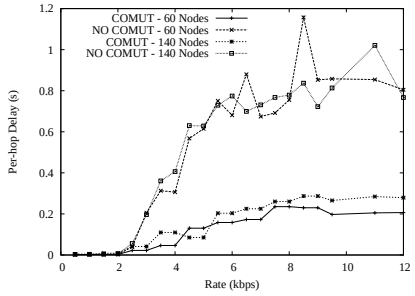

Figure 6. Avgerage Per-hop Delay vs Rate (High & Low Density).

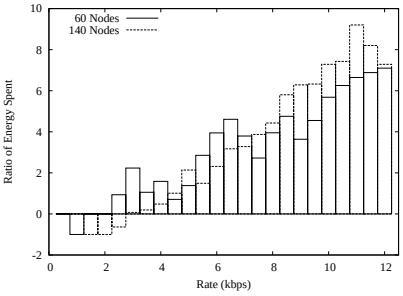

Figure 7. Energy Dissipation Ratio vs Rate.

rate. We observe that for low rates, the ratio is negative, indicating the total energy due to wasteful transmissions and due to control overhead is higher with COMUT. Clearly, in the absence of congestion, the control overhead is wasteful and this is a direct artifact of this effect. Note, however, that the number of control overhead packets is quite low in these scenarios (Fig.8). As we increase the rate, COMUT facilitates a significant reduction in the number of wasteful transmissions and we observe that this more than compensates for the expended overhead.

In Fig.8, we plot the number of overhead messages (sent by the sensors to the sentinel and exchanged between cluster sentinels) as a function of the network density. When the network density increases, the number of sentinels also increases. However, the number of messages does not increase significantly. This is because the update process is triggered only if an intensity threshold is exceeded. Thus, there are a restrictive set of conditions that will have to be satisfied in order for the updates to be generated.

5. Related Work

In this section we review related work on congestion control and prioritization in sensor networks.

CODA [15] is a congestion detection and avoidance scheme for sensor networks that combines local backpressure techniques and sink-to-sensors notifications but is not specifically concerned with different classes of packets. A more recent work [9] proposes techniques that are similar to those in CODA while providing detailed cost metrics in the context of realistic workloads. Protocols such as CODA may extend their *reactive* backpressure techniques used in order to stall low priority packets. These schemes do not prevent congestion and, thus, are not resistant to sensitive drops of high priority data.

In [4], a transport layer solution is proposed for achieving fairness in terms of the number of packets sent by each sensor. This solution assumes a tree structure, dividing the achievable rate among the children nodes. Our work differs in that we target sensor coordination at the cluster level, which considers interactions among multiple, distinct flows. ESRT [13] is another transport layer solution which regulates the sensors' rate so that reliability (with respect to the number of packets received) is achieved while avoiding congestion. ESRT, however, is better suited for monitoring sensor networks that report values periodically.

There has been some work on priority support in ad hoc wireless networks. In [19], an admission control and rating policy mechanism for real time and best effort traffic in a wireless ad hoc network is proposed. Unlike COMUT, in [19], a real-time flow may not be admitted because it may degrade the throughput of other existing real-time flows. Another approach is QoS scheduling for bursty flows proposed in [20] that allows high priority flows to claim bandwidth assigned to low priority flows. However, this would require the use of TDMA which is difficult to enforce in

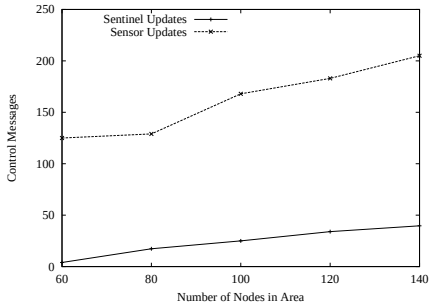

Figure 8. Overhead Control Messages vs Network Size.

sensor networks.

MAC layer prioritization schemes such as [18] provide extensions to the IEEE 802.11 protocol to allow for higher priority packets to be sent out from a local queue ahead of low priority packets [12, 16, 18]. However, even with these schemes, due to interference range effects, low priority packets contend with those of high priority. Furthermore, loss of control packets still could occur causing route failures and other associated effects.

SPEED and RAP are routing transport layer protocols that use speed along the transmission path as the priority metric to distinguish flows [8, 12]. These protocols do not consider the interactions of multiple flows with statically assigned priorities. These protocols resort to backpressure when congestion occurs. Furthermore, they require location information that could be expensive.

6. Conclusions

In this work we propose a framework for supporting flows that have multiple associated levels of importance, in sensor networks. The key objective is to provide high service quality to flows of high importance even under conditions of congestion. Our framework is based on the formation of sensor clusters which cooperate to proactively compute and appropriately disseminate information with regard to the observed network traffic intensity. This allows source sensor clusters to appropriately adjust their rates in response to varying congestion levels. Our simulations demonstrate that our framework is highly successful in abating congestion and in reducing wasteful packet drops, achieving energy savings. Packets from flows of high importance are delivered with extremely high fidelity. We show, via simulations, that congestion is controlled and all important flows can be admitted and delivered with minimum drops, achieving energy savings. For our future work, we intend to experiment with a larger number of priority levels for the flows.

Acknowledgments

This research was supported by NSF Award #0330481.

References

[1] Crossbow MICA motes. Available at www.xbow.com.

[2] NS–2. Available at www.isi.edu/nsnam/ns.

[3] K. Akkaya and M. Younis. A survey on routing protocols for wireless sensor networks. *Elsevier Ad Hoc Network Journal*.

[4] C. T. Ee and R. Bajcsy. Congestion control and fairness for many-to-one routing in sensor networks. In *Proc. ACM SenSys*, 2004.

[5] J. Elson and D. Estrin. Time synchronization for wireless sensor networks. In *Proc. of IPDPS*, page 186, 2001.

[6] Z. J. Haas and M. R. Pearlman. The zone routing protocol for ad hoc networks. Internet Draft, 1998.

[7] J. F. Hayes and T. Babu. *Modeling and Analysis Of Telecommunication Networks. 2nd edition.* Wiley-Interscience, New Jersey, NJ, 2004.

[8] T. He, J. A. Stankovic, C. Lu, and T. F. Abdelzahe. SPEED: A stateless protocol for real-time communication in sensor networks. In *Proc. of ICDCS*, 2003.

[9] B. Hull, K. Jamieson, and H. Balakrishnan. Mitigating Congestion in Wireless Sensor Networks. In *ACM SenSys*, 2004.

[10] J. Kurose and K. Ross. *Computer Networking: A Top Down Approach Featuring the Internet, 2nd edition.* Addison-Wesley, 2002.

[11] S. Lee, G. Ahn, X. Zhang, and A. T. Campbell. INSIGNIA: an IP-based quality of service framework for mobile ad hoc networks. *Parallel Distributed Computing*, 60:374–406, 2000.

[12] C. Lu, B. M. Blum, T. F. Abdelzaher, J. A. Stankovic, and T. He. RAP: A real-time communication architecture for large-scale wireless sensor networks. In *Proc. IEEE RTAS*, 2002.

[13] Y. Sankarasubramaniam, O. Akan, and I. Akyildiz. ESRT: Event-to-sink reliable transport in wireless sensor networks. In *Proc. of ACM MOBIHOC*, 2003.

[14] C. Y. Wan, A. T. Campbell, and L. Krishnamurthy. PSFQ: A reliable transport protocol for wireless sensor networks. In *Proc. ACM MOBICOM*, 2002.

[15] C. Y. Wan, S. B. Eisenman, and A. T. Campbell. CODA: Congestion detection and avoidance in sensor networks. In *Proc. ACM SenSys*, 2003.

[16] A. Woo and D. E. Culler. A transmission control scheme for media access in sensor networks. In *Proc. MobiCom*, pages 221–235, 2001.

[17] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proc. ACM SenSys*, 2003.

[18] X. Yang and N. H. Vaidya. Priority scheduling in wireless ad hoc networks. In *Proc. of ACM MOBIHOC*, 2002.

[19] Y. Yang and R. Kravets. Throughput guarantees for multi-priority traffi c in ad hoc networks. In *Proc. of IEEE MASS*, 2004.

[20] H. Zhu and G. Cao. On improving service differentiation under bursty data traffi c in wireless networks. In *Proc. IEEE INFOCOM*, 2004.