Cluster–based Congestion Control for Sensor Networks*

KYRIAKOS KARENOS,
VANA KALOGERAKI
and
SRIKANTH V. KRISHNAMURTHY

Department of Computer Science and Engineering
University of California, Riverside

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 the flows to experience high packet loss and arbitrary packet delays. This is particularly problematic in event-based sensor networks (such as those in disaster recovery missions) where some flows are of greater importance than others and require a higher fidelity in terms of 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 where, each data source, depending on the relative importance of its data flow and the experienced congestion en route the sink, is coerced into controlling its rate. Our simulation results demonstrate that (i) our techniques are highly effective in dealing with multiple, interfering flows and in achieving high delivery ratios and low delays compared to traditional approaches, (ii) operate successfully over multiple underlying routing protocols, (iii) provide higher throughput to higher importance flows (iv) are responsive to failures and, finally, (v) achieve substantial energy savings due to the considerable reduction in packet drops via the effective regulation of the network load.

Categories and Subject Descriptors: C.2.2 [**Computer Communications Networks**]: Network Protocols—*congestion control*

General Terms: Algorithms, Design, Experimentation

Additional Key Words and Phrases: sensor networks, rate control, clusters, real time

1. INTRODUCTION

In this paper we present a scalable and distributed framework for alleviating 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. Infor-

*This research has been supported by NSF Award 0330481. This is an extended version of our paper published in IEEE EmNetS-II [Karenos et al. 2005].
Author's Address: Department of Computer Science and Engineering, Engineering BU2, Room 351, University of California, Riverside, CA 92521

mation from a multiplicity of sensor-detected events, in such scenarios, will typically have to flow via data paths that are largely interference-coupled, towards the sinks. Collisions of packets from simultaneous 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.

A central requirement in tackling the problem of congestion control is *effective* congestion detection. Congestion can become apparent as network nodes consistently drop packets. Clearly, since our objective is to be able to provide a high fidelity in terms of high delivery ratios, an effective congestion detection mechanism should monitor the factors and conditions that result in packet drops in order to infer the possible onset of congestion.

Packet drops are mainly attributed to queue overflows and media access contention[1]. The problem of queue overflow is even more critical in sensor networks where queue size limitations are substantial. Overflows are caused by disparate flows (initiated at different locations in the network) that follow routes that converge at some common node, overwhelming its incoming buffer. Even if the flows have moderate transmission rates, the combined incoming rate to a common routing point might cause the buffers to overflow. Even with congestion aware routing schemes [He et al. 2003], it is still very possible that flows will converge in the proximity of the sink. At the same time, media access contention can also cause packet drops. As contention grows, the waiting time for obtaining the channel for transmission also grows. As a node waits for the channel, additional traffic may arrive filling up its queue and further increasing the packet delays.

To infer congestion, previous research on sensor network congestion detection has considered localized techniques such as queue size [Sankarasubramaniam et al. 2003] or MAC-level statistics [Wan et al. 2003]. However, it is possible that neither of the two methods can effectively identify or predict a congested network. On the one hand, sensors have small queue sizes, which, even with low contention might overflow. On the other hand, a queue might be highly utilized, and yet may not necessarily suggest congestion. Therefore, a metric that can capture the effects of both queue occupancy and contention while being computationally simple for application on sensor devices, is required. Furthermore, we should be able to combine partial and local congestion assessments to generate congestion estimates that span an area beyond the local scope of each node. This would provide more accurate, distributed and timely feedback concerning the network and help stimulate appropriate controls at active as well as prospective sources.

Traditional congestion control approaches utilize end-to-end or hop-by-hop (or combinatory) control techniques [Wan et al. 2003; Ee and Bajcsy 2004]; however, these prior approaches consider only a single class of packets. End-to-end techniques [Sankarasubramaniam et al. 2003] require the sink to regulate the sensors' sending rates. However, because one may expect the traffic volume to be high in the proximity of the sink, the regulatory updates sent by the sink may be throttled at the source. Furthermore, end-to-end techniques might not provide responsiveness in a timely fashion, especially when highly unpredictable events occur. On the other hand, hop-by-hop, backpressure techniques [Wan et al. 2003] are reactive in nature. Backpressure messages are sent when congestion has already occurred and this may cause a significant initial loss of packets while the control messages are en

[1]Channel errors may cause packets to be dropped if they are erroneously received. However, this situation is relevant to the link reliability problem, which we do not address herein. In this work we first consider a reliable channel and attempt to tackle the problem minimizing packets drops that are solely attributed to congestion. Later we investigate the impact of stochastically varying, unreliable channel on the performance of our congestion control scheme.

route the sources. An intermediate action in which packets are delayed by storing them temporarily, is not practical in sensor networks due to the limited memory size. Therefore, with backpressure, there is no choice other than to drop packets en route, in order to reduce congestion [He et al. 2003].

While alleviating packet drops by means of congestion control, another challenge is to provide preference to packet flows of higher importance. More specifically we would want to provide higher throughput to important flows (*i.e.,* allow for a higher number of important packets in the network compared to concurrent flows of lower importance). Previously proposed service differentiation techniques have not considered congestion or its effects [Yang and Vaidya 2002; Lee et al. 2000]. Admission control techniques proposed for wireless, ad hoc networks [Lee et al. 2000; Yang and Kravets 2004] regulate the network load and, thus, control congestion indirectly. However, these methods are likely to be computationally intensive in the presence of multiple classes, hence, are unsuitable for sensor networks.

In this paper we present COMUT (COngestion control for MUlti-class Traffic), a framework that consists of 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 the network and collect statistics to infer the *collective* level of congestion. Regulation of sensor rates (*per-cluster*) and coordination between cluster nodes is achieved by exchanging only small amounts of control information (via control packets) between the sentinel sensors along flow paths. The main benefit of sensor clustering is that a group of sensors can capture the behavioral interactions between flows. Our approach improves responsiveness in comparison to end-to-end techniques, while, expensive (in terms of communication overhead), reactive hop-by-hop backpressure methods are avoided. COMUT generates and propagates rapid and accurate feedback about active flows to source sensor clusters to signal congestion. 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. This process improves the timeliness of data delivery seen by flows of high importance. The efficiency with which the available bandwidth is shared between flows is also improved.

Contributions: 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:

—A distributed and scalable mechanism that facilitates the clustering of sensors and allows for the adjustment of the sending rate per cluster.

—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: (i) our techniques are highly effective in dealing with multiple, interfering flows and in achieving high delivery ratios and low delays compared to traditional approaches, (ii) operate successfully over multiple underlying routing infrastructures, (iii) succeed in providing higher throughput to higher importance flows, (iv) are responsive to failures and unreliable links and, finally, (v) achieve substantial energy savings due to the considerable reduction in packet drops via the effective regulation of the network load.

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 scattered 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. The interference coupling between paths becomes pronounced in regions close to the sinks. Typical routing protocols build hop-by-hop paths to the sink and the sources are unaware of the existence of other flows in the network. One might attempt to perform congestion aware routing [He et al. 2003]. In [He et al. 2003], sensors avoid overloading congested areas by redirecting traffic away from hotspots. However, in disaster rescue missions, congestion can vary dynamically and there is overhead in reconstructing the routes and maintaining state information at each node. The problems are further exacerbated in the presence of flows belonging to multiple classes, since information regarding 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, then, to give preference to and maximize the throughput of flows of higher importance.
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. This dynamically changes the interference patterns among flows and hinders our ability to predict network 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 and, thus, reactivity does not avoid critical packet losses. Our goal is to *proactively* monitor the network and identify or predict the onset of congestion. 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.
The need for Cluster-based Traffic Intensity Estimation: Given that we need an efficient, proactive rate control scheme, it is essential 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. Furthermore, in order to reduce the traffic load without 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 by means of network control packets. However, this would be energy intensive and will unnecessarily increase the network traffic and, thus, will further contribute 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 examining the convergence of traffic from multiple flows and thereby produces a more accurate representation of flow interactions.

3. CLUSTER-BASED NETWORK STRUCTURE

As stated, our mechanisms are employed on per cluster bases. Clustering is a technique for grouping together a number of nodes. Usually, one node among the group members is elected to be the group leader or cluster–head. The cluster–head plays the role of the central coordinator depending on the task that the clustering process is designed to support. Clustering in ad–hoc networks has been applied for routing [Wu 2002; Haas and Pearlman 1998; Krishna et al. 1997], load balancing [Amis and Prakash 2000] and for supporting security [Bechler et al. 2004]. In sensor networks it was shown that clustering for data collection [Heinzelman et al. 2000]

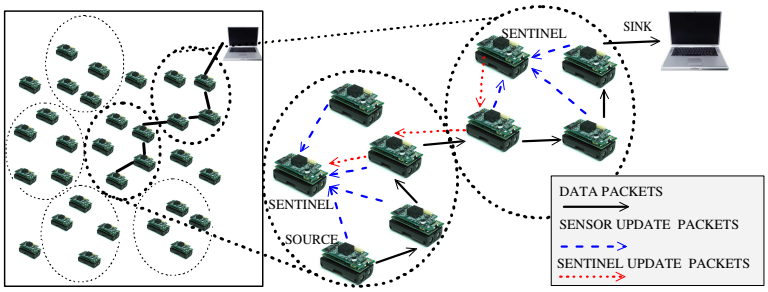


Fig. 1. An Overview of COMUT Network Structure.

and data aggregation [Mhatre and Rosenberg 2004] is highly effective in conserving energy and prolonging the network lifetime.

In COMUT, however, the objective of clustering is to achieve *effective congestion estimation and rate control*. Sensors utilize cluster–level (localized) network monitoring to reflect the traffic load within a local spatial region. Each sensor cluster is governed by an appointed *sentinel* (*i.e.,* a clusterhead) which is responsible for collecting traffic load updates produced by its cluster members and compiling a collective cluster–level congestion estimate. When such estimates are computed, active or newly initiated flows can be informed whether the regions, via which their packets are routed to the sink, are congested. This information, which is based on the computations of the monitoring processes, is utilized by the cluster sensors in order to control the rates of the flows.

An assumption we make here is that sensor nodes do not go into sleep mode throughout the mission operation time. We assume that in the event-based networks under consideration (such as emergency response and distaster recovery missions) sensor are scattered in the affected area and must be turned on in order to deliver as many – potentially critical – reports as possible.

In Figure 1 we present an overview of the COMUT network structure. Sensors in the network are grouped into clusters, as shown in the figure. Message exchanges consist of regular data packets, intra–cluster load updates (sensor update packets) and inter–cluster sentinel updates (sentinel update packets). Sensors within a cluster periodically report their locally computed estimation of the traffic load (dashed lines). This information is processed by the cluster sentinel and a collective cluster–level load estimates is communicated to the sentinels towards the clusters that the sources belong to (dotted lines). The received value is considered together with the local estimation and a new estimate is produce which incorporates the traffic conditions *in the vicinity of clusters along the path*, towards the sink. Therefore, sensors at the clusters that generate traffic are able to produce aggregate traffic estimates from the sources to the sink, and thus, adjust their sending rates based on the current congestion level.

Benefits of Clustering: This setup of grouping the sensors into clusters (instead of having each sensor acting independently) provides multiple benefits. Clustering offers scalability while allowing for more accurate local estimations since a collection of sensors in a small region can better capture interactions between multiple flows that pass through that region. In addition, the selection of a single sensor as a representative of the whole cluster, allows for the aggregation of updates into a single cluster–level update to be propagated to the sources. Note that only cluster–head sensors need to exchange control information. This message exchange procedure entails low overhead which, in addition, is justified by the major energy savings that are provided by reducing the packet drops. Dropped packets result in wasted transmission efforts, *i.e.,* energy wastage as shown in Section 7.

Designing the Clustering Algorithm: The basic requirement in COMUT is that each sensor must belong to a cluster in order to be able to communicate its traffic load state. However, the design of COMUT is driven by the following addi-

tional properties: (1) The clustering algorithm must allow for the *self–organization* of the sensors in a distributed manner. (2) The clustering algorithm must have small message complexity during the sentinel election. (3) The resulting number of clusters must be small in order to reduce the update message exchanges among sentinels. (4) Less of a requirement but rather a design choice is the creation of *one–hop clusters* (*i.e.,* cluster members are one hop away from the appointed sentinel). One–hop clusters are considered in most previously proposed clustering algorithms [Lin and Gerla 1997; Heinzelman et al. 2000; Basagni 1999]. One–hop clusters are simpler to create; at the same time, the wireless broadcast advantage is exploited since the sentinel can reach all of its members and notify them of the current congestion level with a single broadcast. In addition, since the probability of dropping a packet increases exponentially over multiple hops [Wan et al. 2002], single–hop messages have a much higher probability of being correctly received. Further, the sentinel can promiscuously listen to packets forwarded by its members and, with no additional overhead, infer the number and the importance level of flows that traverse its cluster.

Clustering Techniques: Clustering algorithms with respect to data communications have been proposed to either produce clusters that satisfy specific properties (such as cluster size, good distribution of cluster–heads etc) or to save energy by periodically re-electing cluster–heads. We are mostly concerned with the first case *i.e.,* algorithms that efficiently build one–hop clusters but can also quickly be re-executed in case of sentinel failures. Proposed algorithms such as [Amis et al. 2000; Younis and Fahmy 2004; Bandyopadhyay and Coyle 2003] can be used while results in [Bandyopadhyay and Coyle 2003] can be consulted to fine tune algorithm parameters. Our algorithm is based on [Amis et al. 2000] and [Bandyopadhyay and Coyle 2003] which provide heuristic methods for constructing d-hop dominating sets, a NP-complete problem [Amis et al. 2000]. The analyses in these works have shown that cluster construction only requires $O(d)$ rounds of message exchanges, while experiments have shown that load balancing is also achieved.

Sentinel Election in COMUT: At the beginning of the *election phase*, 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. Note here that, with CSMA, a sensor cannot send a message when it senses the channel to be busy. Upon the expiry of this time-out, while sensing the channel, if a sensor has received a sentinel announcement message, it will join the sender's cluster and suppress its own announcement. However, if, after the lapse of t_s, no message has been received, the sensor will become a sentinel itself with a probability P_n. P_n is a function of n, where n is the number of iterations during which the sensor has *not* elected itself a sentinel. If multiple announcements are heard during this time period, if n is odd, the receiver sets the sender with the highest *id* as its sentinel else, if n is even, it sets the sender with the lowest *id* as its sentinel. The reason is to allow for sentinels whose cluster members have been overtaken in the previous broadcast, to regain some of their neighboring nodes as members and eventually construct more balanced cluster sizes [Amis et al. 2000]. If a sensor node does not succeed in becoming a sentinel and has not joined a cluster, it increases the probability P_{n+1} of becoming a sentinel at the end of the next time-out, as per the following formula:

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

where, α is a parameter that determines the effective degree of increase of P_n with n. It is important to note that as n increases, the probability of a node becoming a sentinel also increases. Thus, within a small number of steps, each node either becomes a sentinel itself or a member of a cluster, headed by a neighbor sentinel. By carefully choosing the initial probability P_0 and value t_s that is close to the one-hop delay (as argued in [Bandyopadhyay and Coyle 2003]) we can achieve convergence in $O(1)$ (constant) iterative steps as shown in [Amis et al. 2000] regardless of the network size. To provide an intuitive justification for the values chosen, we compute

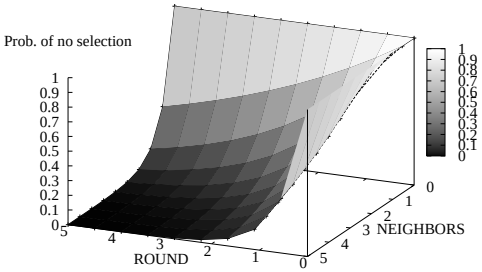


Fig. 2. Upper Bounds of Rounds Required for Sentinel Selection($P_0 = 0.3$).

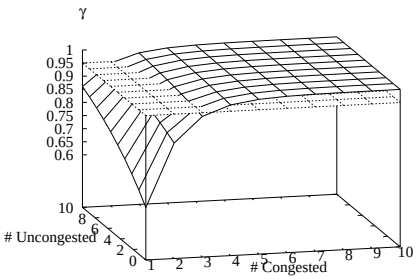


Fig. 3. γ value with variable number of congested sensors.

a theoretical upper bound on the maximum number of rounds required. Consider a set of M nodes that are within the range of each other. If we had not increased the selection probability after the expiry of each timeout, *i.e.*, $\alpha = 0$, the probability that no selection is made after n rounds is $(1-P_0)^{Mn}$ (until the first node transmits a sentinel advertisement). Figure 2 shows this relationship for $P_0 = 0.3$ which we also used in our tests. We can choose t_s to be the sum of the empirical one hop delay and the congestion window at the MAC layer (to minimize possible simultaneous advertisement attempts) which bounds the total required time for sentinel selection to $(n+1) \cdot t_s$.

4. TRAFFIC INTENSITY ESTIMATION

Once the clusters are formed, the objective is to determine the level of local congestion within each cluster; once this estimate is computed, the information is fed back so that the sources of the data flows 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 the nodes in the network and the limitations in terms of communication abilities of sensor nodes. Thus, in such dynamic environments, an exact traffic intensity estimation is very hard to achieve in a practical setting. For these reasons we choose to use an approximate model towards *estimating* the network conditions.

4.1 Traffic Intensity Approximation Model

We use a queuing network model, wherein each cluster is modeled as a queue. The queues are then interconnected to form a network. We assume that the resulting network is a BCMP network of queues [Hayes and Babu 2004]. 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). Our goal is to acquire *macroscopic* network statistics using a heuristic approach that is adequately simple to fit within the computational capabilities of sensor devices. We discuss in Section 4.2 and Section 7, the benefits, accuracy and effectiveness of our approximation. 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; this is defined as $\rho_i = \frac{\lambda_i}{\mu_i}$; λ_i is the aggregate arrival rate of the packets produced and forwarded at sensor i while μ_i is the service rate at sensor i, *i.e.*, $\mu_i = 1/\overline{W}$ where $\overline{W}$ is the computed exponentially weighted moving average of the packets' waiting time at the head of the service queue. The distribution of the queue size $P(k)$ (essentially this is the probability that there are k packets in the queue) at the sensor node is

computed as $P(k) = (1-\rho_i)\rho_i^k$. For N distinct queues, the joint distribution is the product [Hayes and Babu 2004] $P(k_1, k_2, ..., k_N) = \prod_{i=1}^{N} (1-\rho_i){\rho_i}^{k_i}$. We thus, define the *Traffic Intensity* estimate in a cluster, γ (also called the γ estimate), to be the probability that at least one of the sensors in the cluster has a non-empty queue:

$$\gamma = 1 - P(0, 0, ..., 0) = 1 - \prod_{i=1}^{N} 1 - \rho_i \tag{2}$$

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 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 as below:

Let γ_i represent the calculated γ value for cluster i. For cluster i we define the *upstream* clusters to be the clusters from which packets are received by nodes in i, while the *downstream* cluster is defined to be the cluster towards which packets are forwarded from nodes in i. The sentinels of these clusters are correspondingly called *upstream and downstream* sentinels[2].

Let us now consider a specific flow initiated at a cluster s, $s > 0$. Cluster s is called a *source cluster*, *i.e.,* it contains one or more sensors that generate traffic. Let the sink be defined as cluster 0. For cluster 0 we define $\gamma_0 = 0$. Let j be a cluster *along the path* from s to the sink, $s \geq j > 0$. For cluster j (with N members), j receives estimation from a *downstream* cluster $j-1$ (j is an *upstream* cluster for $j-1$). The collective path traffic intensity is calculated by simply incorporating the γ_{j-1} value received at the upstream sentinel j, *i.e.,*:

$$\gamma_j = 1 - \left[(1-\gamma_{j-1}) \prod_{i=1}^{N} 1 - \rho_i \right] \tag{3}$$

We reiterate that the collective calculation of γ is performed per-path, *i.e.*, along the clusters via which a flow under consideration traverses (see also Section 4.2). If paths converge at a single cluster, then the updates that are received by the sentinel from the various member sensors nodes are used to generate an aggregate γ value that incorporates the effects of all the flows through the cluster. The calculated value of γ is used as an indicator of the congestion level. The traffic intensity estimate computed in Equation 2, ensures a high value of γ even if a small number of cluster members are heavily loaded.

We wish to point out here 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 to receive service and, eventually, is sent out to the next sensor node (see also Section 4.2).

We need to define a threshold value γ_{tres} beyond which the cluster is considered to be congested. This will affect when or how often γ estimates will be sent to the source sensors to regulate the traffic. In order to do this, we first consider moderate traffic load conditions where it is reasonable to assume arrivals to approximate the Poisson distribution. It has been shown that, in such networks and under dense connectivity, the *M-M-1* queue provides a highly accurate approximation for each link between nodes [Bertsekas and Gallager 1992]. Furthermore, it has been shown that under the same assumptions, the average number of packets in the queuing system increases very quickly after the queue load reaches a specific threshold [Hayes and Babu 2004]. Based on this theoretical result we have tried to identify a thresh-

[2]A cluster is effectively identified by its sentinel's ID. By *cluster update communications*, we essentially refer to communications between sentinels.

old ρ_{thres} for initiating the update transmissions, that is neither too low (in which case the updates would be useless) nor too high (in which case congestion detection would not be effective). We set this threshold to 0.6 and we present our justifications in the experimental evaluation section. Given this threshold, in a cluster, some sensors may *expect* congestion (*i.e.,* the ρ value exceeds ρ_{thres}) whereas others may experience no congestion. We, thus, assume two quantized levels of congestion: Sensors expecting congestion are associated with a value of traffic load $\rho = \rho_{thres}$ while non congested sensors, with a value $\rho << \rho_{thres}$. We plug these values in Equation 2 and plot, in Figure 3, the value of γ for a variable number of sensors. Figure 3, shows the corresponding value of γ in a cluster for different number of congested nodes (on the x-axis) and uncongested (y-axis) nodes. We find that even with a small number of sensors anticipating congestion the value of γ quickly increases. We chose the value of γ_{tres} to be 0.95 (high enough to indicate congestion) which is reached even when as few as about 3 sensors anticipate congestion. *Note that this is true irrespective of the ρ value observed at the other sensors within the cluster.* In our simulation experiments we observed that this value provided a high degree of accuracy in estimating the traffic intensity for different sizes of clusters. This can also be seen in Figure 3. As shown, in a congested cluster, the deviation of γ for different cluster sizes, is extremely small and closely approximates the value of 1. We reiterate that this value is an estimate: it indicates the probability of finding at least a single packet in a cluster. This estimate is computed at the sentinels of the clusters which contain active or prospective sources to indicate whether active or, respectively, prospective paths towards the sink, are congested.

4.2 On the Properties of the γ Estimator

In this subsection we provide an in-depth discussion on the characterization and properties of the γ estimator. In particular, we provide the motivation for choosing this metric, identify the parameters that affect γ and examine the specific effects.
Capturing the traffic state with a single, distributively calculated value: Equation 2 shows that γ is defined as a function of the packet arrival rate λ_i, the service rate μ_i and the number of active sensors within a cluster (*i.e.,* sensors that send traffic load updates). We examine the properties of γ with respect to these parameters.

First, for the sake of simplicity, let us consider the case wherein only one of the above parameters varies at a time. Assuming a constant service rate μ_i, as the arrival rate λ_i increases, the local traffic load also increases. At some point, the queue of the sensors will be filled up and the overflow packets will be dropped. Thus, with a constant service rate, queue overflow mostly accounts for the effects of congestion. On the other hand, if the arrival rate $\lambda_i$is constant, a reduction in the service rate will cause an increase in the traffic load. Reduction of the service rate is attributed to the delay in servicing packets at the head of the queue due to interference effects among simultaneous flows that contend for the channel. In reality, neither of the two parameters is constant. In our calculation of γ, both queue occupancy and contention are taken into consideration contemporaneously, providing a comprehensive estimate.

Second, in our model we assume that the network of queues approximates a BCMP network, wherein the steady–state probability distribution can be represented as the product of the probability distributions of the distinct queues in the network, studied in isolation. This allows us to compile the partially (per-cluster) calculated traffic intensity values into one collective traffic intensity value. This model although approximate, lends simplicity and tractability and is adequate for our purposes of acquiring *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 in Section 7. In summary, the merits of choosing this metric are twofold:

—*The γ estimator captures the joint effects of both queue occupancy and channel*

contention in a single high level parametric value.

—*Locally calculated, cluster-level γ values can be combined to provide a collective estimate of the congestion level across a larger deployment area.*

Two valid observations can be made here. The first observation is related to the interference effects caused by sensors that are active but do not belong to the same cluster. Our estimate captures this effect since the resulting ρ that is collectively reported by the sensors to their sentinel in the cluster incorporates the delay in service time that was experienced due to the contention with neighboring sensors. The second observation is that due to congestion it is possible that dropped packets will cause the incoming packet rate to a forwarding sensor to be reduced, thus affecting (reducing) the resulting estimation of ρ. This would suggest that due to the reduced input rate, no congestion is experienced in the cluster. This case can still be handled correctly since congestion will be detected at the sensor that actually experiences drops. This latter effect will cause the γ value in the corresponding cluster to be high, and this in turn, effects the collectively estimated value of γ.

Per path update propagation and aggregation: Although our approximation model is valid for network-wide congestion estimates, we chose to apply it only for *per-path* estimates. The reason is that our goal is to reduce drops of important packets by attacking congestion at its origin, *i.e.*, by controlling the rate at the sources. Paths are created from these sources to the sink and therefore a source is only interested in knowing the condition of the network along the path that its packets will traverse. It, therefore, makes sense to send updates only along these paths and more specifically, *upstream*, towards the sources. By aggregating cluster reported values into a single value allows us to reduce the number of updates that need to be sent upstream. In addition, a control packet needs to only reach the sentinel of each cluster along the path. The sentinel will incorporate all received values and provide the collective estimate to all the sensors in the cluster via a one-hop broadcast. We note that, in the worst case, the paths may ultimately converge at the sink. Even in this case, the interference between flows will eventually be taken into consideration.

5. RATE REGULATION

Once the traffic intensity along a path is estimated, our objective is to regulate the rate at which source sensors will send their data toward the sink.

Initialization: Consider the sensor network before any packets are generated. Since flows are yet to be initiated, sentinels do not have any knowledge of where updates are to be sent. Flooding updates to all sentinels within a preset neighborhood is wasteful in terms of energy. Since traffic flows from the source clusters to the sink, sentinels need to only send updates to upstream clusters, *i.e.*, towards 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.

An additional requirement is that before prospective sources can start sending packets, they must have an estimate of how long they should wait between packet transmissions, for an update from the sentinel in their cluster that indicates the congestion level along the path of the flow. The sentinel, would have to compute this based on information exchange with the other sentinels on the *downstream* path toward the sink. *This information is needed by the sources to decide upon the amount 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)*. Consider the packets sent by the sentinels in the previously described path estimation process. RRE is computed at the sink by calculating the total

delay experienced by a packet[3]; 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. As discussed earlier, in order to save energy, a sensor transmits its computed value to the sentinel only if it exceeds a preset 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 that is followed by each flow passing through the cluster. This provides an indication of the level of congestion in the vicinity 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 I).

Although our techniques are designed with an objective of reducing the number of control messages and thus the energy consumption, periodic broadcasting in sensor networks clearly incurs energy expenditure. However, broadcasting is inherently deployed in many existing sensor systems (*e.g.,* periodic beaconing in motes to discover "good" forwarding neighbors to the sink [Woo et al. 2003]; in SPEED [He et al. 2003] periodic beaconing is used to keep track of per-hop delays). In our experiments, we show that the energy consumed due to this communication overhead is in fact compensated by saving energy that would otherwise be consumed due to wasteful receptions and forwarding of packets that are eventually dropped en route the sink, 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 an adaptive scheme that is applied to each flow and follows a regulation policy that is similar to the popular Additive Increase, Multiplicative Decrease (AIMD) policy with TCP [Kurose and Ross 2002][4]. In our technique, however, the rate is dropped to a minimum, *minrate*, for packets of low importance (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 determined based on a few initial experiments of via simulations and can be set before the deployment is in full–fledged operation.

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 indicates that higher importance events should be sent with a higher rate. 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. As

[3]While computing this one-way delay, we assume sensor synchronization, achieved with techniques such as in [Elson and Estrin 2001].

[4]Note that queue management techniques such as Drop–tail and RED are not considered in this work. The main reason militating against utilizing such alternatives is the fact that they enforce packet drops to reduce congestion (which is undesirable both because of energy wastage as well as possible drops of important packets). Our goal is to provide transparent rate control *at the sources* so as to eliminate congestion (and consequently packet drops and delays) at their origin.

a result, our scheme anticipates the injection of dynamic flows in the network and proactively regulates the rate while waiting for congestion feedback.

6. SENSOR FAILURES

In sensor networks, failures (or, in general, unavailability of sensor nodes in the network) are inevitable due to a number of reasons. First, sensors usually operate on batteries which limit the device's operational lifetime.Typically, batteries cannot be recharged or replaced immediately after they are drained; thus battery–depleted sensors cannot be part of the network during that period. Second, the sensor devices are usually put to operation in harsh environments and may actually suffer physical damage. Finally, energy saving schemes that are based on sleep cycles are widely utilized [Woo and Culler 2001; Heinzelman et al. 2000], resulting again in temporary or prolonged sensor unavailability.

A sensor failure significantly affects the routing protocol. Routing protocols have dealt with node failures in various ways. Although dealing with failed nodes at the routing layer is beyond the scope of this work, we are interested in looking at how COMUT is affected by failed nodes and route repairs. AODV, for instance [Perkins and Royer 1999], reactively builds routes on–demand, thus, the responsibility of repairing a route is left to the source which makes a new route request when it receives a route failure indication. ZRP [Haas and Pearlman 1998], on the other hand, utilizes proactive messages to invoke repairs, i.e., nodes within the local routing zone are updated periodically. In SPEED [He et al. 2003], sensors learn about dormant or dead sensors by means of the periodic beaconing process, as in ZRP. However, unlike ZRP, knowledge dissemination is only limited to a node's one-hop neighborhood. Since SPEED is based on Geographical Forwarding, it takes routing decisions at each hop en route the destination. Thus, SPEED resorts to *rerouting* (*i.e.,* forwarding packets to another sensor towards the destination) in order to deal with failed nodes. The effects of route repairs on COMUT are examined in the following subsection.

6.1 Cluster Member Failures

A failed sensor may be either a cluster member or a sentinel. It is important to differentiate between sentinel failures and non-sentinel failures. We, next, investigate these two cases separately.

If a cluster member fails then different cases need to be considered:

(1) If the sensor does not forward or generate traffic then there will be no effect on COMUT. The sensor does not send updates therefore it does not contribute to the calculation of γ.
(2) If, however, the sensor produces traffic then it will seem as if the sensor has stopped executing its programmed task, in which case an application-specific repair process must be initiated.
(3) If the sensor forwards packets, a route failure will occur and the routing protocol would act as per its specifications.

We first consider a case where a routing protocol invokes *rerouting* upon a node failure. When a sensor fails it is required by the remaining sensors to forward additional (a higher number of) packets. This will cause an increase in the traffic load of these remaining sensors which will consequently result in these sensors reporting higher ρ values. Ultimately, the cluster γ value will increase. Note that if this increase leads to the γ exceeding the congestion threshold, COMUT rate control will naturally be applied at the sources.

Next we consider a case where the routing protocol relies on *route repairs* (such as in AODV). If traffic was being forwarded by a failed node, a route failure is signaled. While the routing protocol reconstructs the paths, COMUT may again perceive higher loads within the affected clusters because of the introduction of the additional *route–repair* control packets. The rate regulation process will be triggered which will coerce the sources into reducing their sending rates. The

reduction of the network load will assist in faster route restoration while fewer, or even no packets will need to be dropped due to the unavailability of a forwarding node.

Since COMUT relies on the underlying routing protocol to deliver its control messages, one can claim that a route failure can occur while forwarding such control packets (*i.e.,* γ updates). In this case, COMUT rate control might not be engaged. However, we argue that the design of COMUT provides an inherent robustness by ensuring that the distance in hop count between any two neighboring sentinels is relatively small, as we deliberate below. The shorter the paths, the less likely they are to fail. Furthermore, shorter paths can be repaired in a fairly short time-span, if they were to fail.

6.2 Sentinel Failures

COMUT is directly affected by sentinel failures. The failure of a sentinel means that the collective Traffic Intensity cannot be computed before a new sentinel is selected. The failure of a sentinel can be identified by the cluster members via the routing protocol's neighborhood maintenance process (supported by some routing protocols such as ZRP). An example is using periodic beaconing. As soon as such detection occurs, the election process is started. The process is the same as described in Section 3. A sensor receiving a sentinel announcement message joins the cluster by simply replacing the value of its cluster ID in the messages it broadcasts.

However, it is not always the case that the routing protocol supports neighborhood maintenance. In such cases sentinel failure detection can be realized as soon as a sensor starts broadcasting its ρ values. If the MAC layer supports acknowledgments (such as in 802.11 CSMA) a sensor can detect failure to transmit to the sentinel. Otherwise, a sensor can reset its sentinel (i.e. become part of another cluster) if it fails to receive sentinel updates for a specific period of time which can be set to double the update period. Note that, since the sentinel is only one hop away, it is highly possible that sentinel updates are received with high probability, as discussed in Section 3. If no near–by sentinel is available, the sensor (as per the selection algorithm) will become a sentinel itself. With respect to γ updates, if a sentinel is lost, these updates will not be computed and propagated and no regulation takes place. However, as soon as a new sentinel is elected, after detecting the failure as discussed above, the traffic intensity computation procedure resumes normally.

7. EVALUATION

We have evaluated the performance of COMUT by means of simulations performed with the Network Simulator (NS) tool [ns].

Simulation Settings: We generated random network topologies in an square area of $100m$ x $100m$. We have constructed *sparse* networks by randomly and uniformly scattering 60 sensors in the square area and *dense* networks by placing 140 sensors in the same area. One of the sensors was selected at random to be the sink. A CSMA–based MAC protocol with an exponential backoff policy was assumed and RTS/CTS exchanges were enabled; the use of this MAC protocol has been generally assumed in prior work on sensor networks [Sankarasubramaniam et al. 2003; Wan et al. 2002; Wan et al. 2003] that do not specifically address the MAC layer (as with our work). However, since we target aperiodic-type applications with multiple simultaneous bursts (such as disaster response) we assume that sleep cycles are not utilized as they would be in the case of monitoring applications.

The sensor nodes are set up to simulate the characteristics of MICA motes [motes]. They are homogeneous and have a transmission range of a *maximum* of $25m$ in free space. Data packets are 30 bytes (the size as per the standard specifications). The memory in a mote is 4KB; thus, the queue size is restricted to accommodate at most 65 packets. In order to isolate the effects of congestion, we first assume the

Number of Sensors	60 (sparse), 140 (dense)	**Network Bandwidth**	40 *kbps*
Area	$100m \times 100m$	**Data Packet Size**	30 bytes
Transm. Range	25m	**Minimum Rate**	10 p/sec
Queue Size	65 packets	**Burst Duration**	30sec

Table I. Parameters Used in Simulations.

presence of robust physical layer techniques (FEC codes) to cope with bit errors[5]; hence, packet drops are attributed only to queuing related drops and to collisions. We simulated a scenario where *four* concurrently occurring events are sensed at random points in the sensor field (an *event burst*). The flows interfered with each other at all occasions, especially at the proximity of the sink. Stress–testing the schemes was achieved via increasing the sending rates. Events are assigned either a *high* or a *low* importance level. Additional parameters used in our simulations are listed in Table I.

Architecturally, COMUT is placed above the routing layer. We have implemented COMUT on top of three different routing protocols. Each of these protocols follows a different route discovery approach: *On Demand routing* (AODV [Perkins and Royer 1999]), *Hybrid Reactive/Proactive Routing* (ZRP [Haas and Pearlman 1998]) and *Stateless Routing* (Geographical Forwarding). AODV [Perkins and Royer 1999] is an *on demand* routing protocol, *i.e.,* it builds routes to destinations only when requested by the source. The Zone Routing Protocol (ZRP) [Haas and Pearlman 1998] is a routing protocol that combines both *proactive and reactive routing* approaches. Proactive information is maintained within (2-hop) neighborhoods, referred to as *zones*, that are constructed around each node. Finally Geographical Forwarding (GF) is a greedy routing algorithm which tries, at each hop, to forward packets closer to the destination in terms of actual distance.

As shown in Table II, we implemented and compared COMUT to a basic (BASE) scheme, a reactive backpressure control scheme (BP-AIMD) and a rerouting scheme (REROUTE). In the basic scheme, BASE, no congestion control technique is utilized. The BP-AIMD scheme is a reactive backpressure control scheme where queue overflow indicates congestion and AIMD was used for rate control. In the REROUTE scheme , congestion was handled by rerouting the packets. We have implemented rerouting in conjunction with Geographical Forwarding as proposed in [He et al. 2003], where the SPEED routing protocol for real time sensor networks is proposed. In SPEED, greedy geographical forwarding is used to forward packets towards sensors, only when they can maintain a required delay bound. Congestion is detected when high delays are experienced while forwarding packets. Backpressure notifications are then sent by the congested sensor. The upstream receiver of the backpressure notification then reduces the probability of forwarding packets to the congested sensor and attempts to send them via another neighbor (reroute). SPEED resorts to packet drops in order to reduce the forwarding traffic load and reduce the experienced delay at the downstream sensors.

In our protocol's evaluation we have conducted three extensive sets of experiments. In the first set of experiments we take a microsopic view of the operation of our technique, justify the selection of the parameters and present how the γ estimator accurately identifies a congested cluster. In the second set of experiments, we compare COMUT with BASE, BP-AIMD and REROUTE schemes over multiple routing protocols. We also evaluate the performance of COMUT when node failures occur. Finally, we illustrate the performance merits of COMUT under more realistic channel effects.

7.1 Microscopic COMUT Experiments

Selecting ρ_{thres} and Update Interval: Choosing the right update transmission threshold is important. The value should not be too low to cause wasteful updates but at the same time not too high to prevent congestion from being identified in its

[5]Later (Sec. 7.3) we consider unreliable links and their effect on COMUT.

	Congestion Detection	Notification	Rate Control
BASE	None	None	None
BP–AIMD	Queue Overflow	Bckpressure	AIMD
REROUTE	Transmission Delay	Rerouting	None
COMUT	γ Value	Proactive	COMUT

Table II. Techniques for Congestion Detection, Notification and Reaction

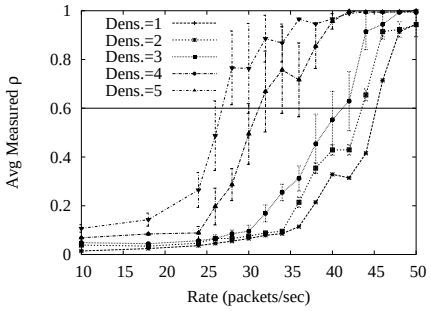


Fig. 4. ρ Values for Variable Transmitting Nodes Density.

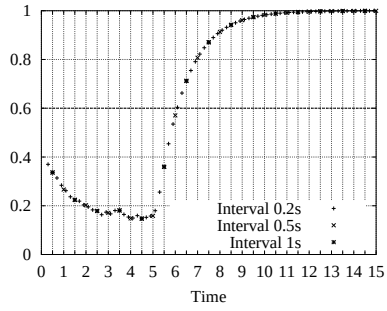
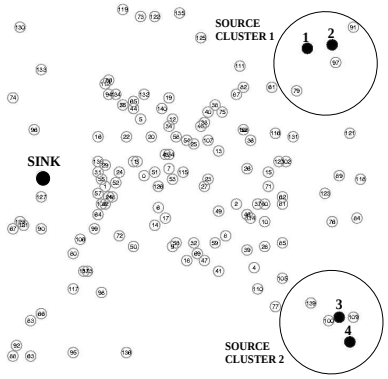


Fig. 5. Accuracy of the Update Interval.

early stages. In Figure 4 we plot the range of values for ρ as measured for different number of transmitting nodes within a cluster. The first observation is an increase in the variation of the measured values around the value of 0.6, an indication of an unstable state. Second, as expected from the analyses in [Hayes and Babu 2004], we observe that beyond the 0.6 value, the load increases very quickly and grows faster as the number of contending nodes increases. This continues until, eventually, all nodes are overloaded. As shown in Figure 3, even when this value is reached by a small number of nodes it serves as a proactive forewarning, indicating a congested cluster; we revisit this property later in our evaluation.

In Figure 5 our goal is to illustrate the selection of the interval at which ρ is recorded and sent to the sentinel. We start sending packets with a low rate initially (*10 p/s*). At 5 seconds of simulation time, we increase the rate to 40 *p/s* – a high enough rate to produce congestion as observed in the ρ selection test. ρ is computed at every packet arrival instance and show the values recorder at intervals of 0.2, 0.5 and 1 seconds. We are studying the sensitivity of our approach to the frequency at which ρ is measured. In particular, we want to ensure that a transition from a stable (uncongested) state to an unstable state (i.e. the onset of congestion) is immediately recognized. This transition was observed to be approximately *1 sec*, thus a 0.5 *sec* interval is adequate to capture it early on and initiate the sentinel update process at regular intervals of this duration.

Benefiting from Proactive Clustered Monitoring: Next, in order to demonstrate the benefits of clustering we construct a network with a non-uniform distribution of nodes. More specifically, we consider a network in which certain areas are densely packed with nodes. These dense regions carry flows that are interference coupled within the same clusters (Figure 6). We used two source clusters of two sending sensors each, a total of four flows, and set the maximum rate to 40 *p/s*. As sensors increase their rates, as per the additive increase policy, the contention in the commonly used dense area increases, exacerbating the effects of congestion, and more particularly, packet drops. In Figure 7 the number of dropped packets is shown. We observe that contention is moved backwards towards the sources due to transmission delays in the dense area. While the throughputs and delivery ratios are similar with BP-AIMD and COMUT (we omit these results due to space limitations, however delivery ratio and throughput for various routing protocols will be illustrated in detail in the following subsections), the rate of drops was lower



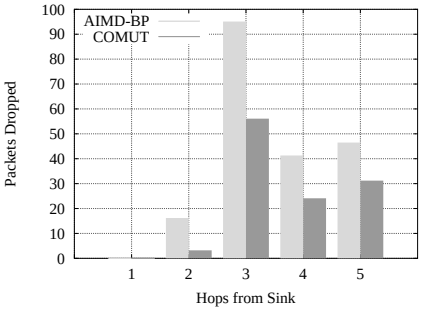


Fig. 6. Non-uniform Topology Fig. 7. Dropped Packets Vs Hops.

for COMUT. The reduction in the rate of drops is attributed to the following effect. With COMUT, multiple nodes in the region of contention, reach the ρ_{thres} of 0.6 *prior* to experiencing large numbers of drops. Then, they quickly indicate the onset of congestion to the sources, which in turn, drop their rates. Note that the wasted energy due to the dropped packets that have already traversed a large number of hops is considerably reduced with COMUT. We support this with additional experimental results described in the following section.

Effects on Fairness: While the goal of COMUT is *not* specifically to support fairness among flows, (on the contrary one of its goals is to ensure higher throughput to more important flows, as illustrated later) we will study the effect of COMUT on this metric. Fairness is a function of the flow paths and the MAC protocol in use. We have chosen the flows such that the number of hops traversed is the same (*5 hops*), thus, we minimize the effects of the routing path on the performance enjoyed by the flows. We observe that due to the cluster–based nature, the flows are collectively notified to drop their rates; thus fairness is not harmed as shown in Figure 8. We note that recent works have concentrated more specifically on the fairness aspects of rate control [Ee and Bajcsy 2004; Rangwala et al. 2006].

Evaluation of the γ Estimator: In the next set of experiments we evaluated the accuracy with which the traffic intensity estimator (γ estimator) can estimate the traffic congestion in the sensor network. In order to produce controlled sensor network environments, we set up a $200m \times 200m$ grid network containing 196 nodes with a single sink placed above, at the center of one of the sides as shown in Figure 10. 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 (each comprising of a cluster of 3 source nodes) and ensure that they pass through common clusters using AODV. We deterministically identify the clusters via which the flows pass. We plot the value output by the *collective* γ estimator at each of these clusters, with increasing rate, as reported by the cluster sentinel within an observation period of $3sec$.

We present, in Fig. 11, 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 understand 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 ($\approx$ 10 *packets/s*) and *early* during the 3 second observation period; thus, packets are dropped. However, due to delays resulting from congestion, packets fill 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

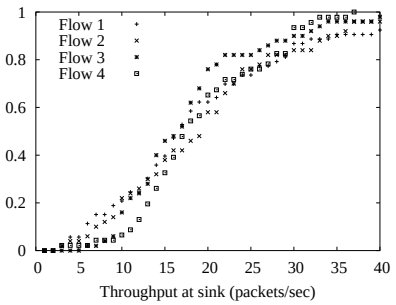


Fig. 8. Examining fairness with COMUT.

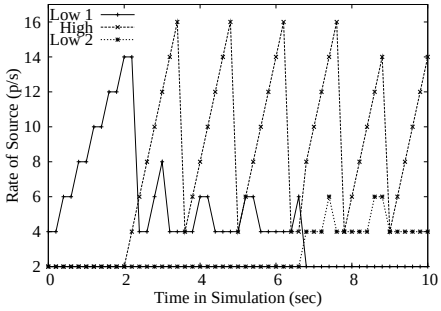


Fig. 9. Time trace for the Sending Rate of Sources.

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 it is not necessary to send updates during stages of low to moderate traffic loads. As discussed earlier, the periodic updates are initiated only if γ exceeds a specific threshold.

System Operations: Our next experiment demonstrates the operation of the system when our framework is in place in the presence of both high and low importance flows. Figure 9 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 0.1 *sec* and the d_p values (Section 5) are 1 for high and 2 for low importance packet flows. Initially, a low importance flow (Low1) is injected at time *zero*. Fig. 9 shows that the rate at which the sources send packets of this flow increase. However, after a high importance flow ("High") is started two 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 pre-specified minimum. Notice that "High" flow increases its rate quickly while "Low1" flow keeps its rate low. When "High" reaches a rate of 16 *packets/sec*, 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 6.6 *sec* we introduce a second low importance flow "Low2"; the flow is intentionally introduced 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 time and location of the *origin* of a flow, our framework succeeds in controlling the rates efficiently.

7.2 Macroscopic Experiments

Delivery Ratio: In the second set of experiments we evaluate and compare COMUT with the BASE, BP-AIMD and REROUTE schemes in terms of delivery ratio, throughput, per-hop delay, energy savings, for both dense and sparse networks. The first important property of COMUT is showcased in Fig. 12. The delivery ratio is defined as the ratio $\frac{Total\ Number\ of\ Packets\ Sent\ by\ all\ Sensors}{Total\ Number\ of\ Packets\ Received\ at\ the\ Sink}$. The figure plots the observed delivery ratio achieved by COMUT with each of the considered routing protocols (AODV, GF, ZRP) for both sparse and dense networks. Note that rerouting is only possible with GF, where the routing process is stateless and, thus, a node may choose to divert a packet to another neighbor in case where the closest neighbor to the destination is congested. The rate indicated on the x-axis reflects the *constant* sending rate with BASE and the *maximum* rate with BP–AIMD and COMUT. Our first observation is that COMUT performs as well as or

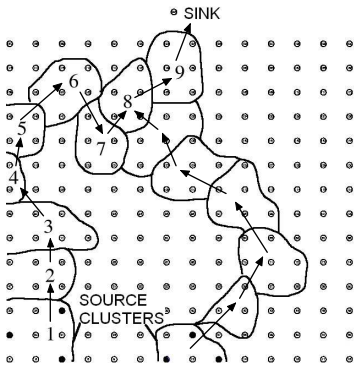


Fig. 10. Paths followed by flows through clusters.

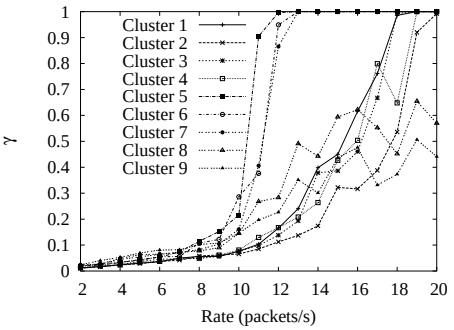


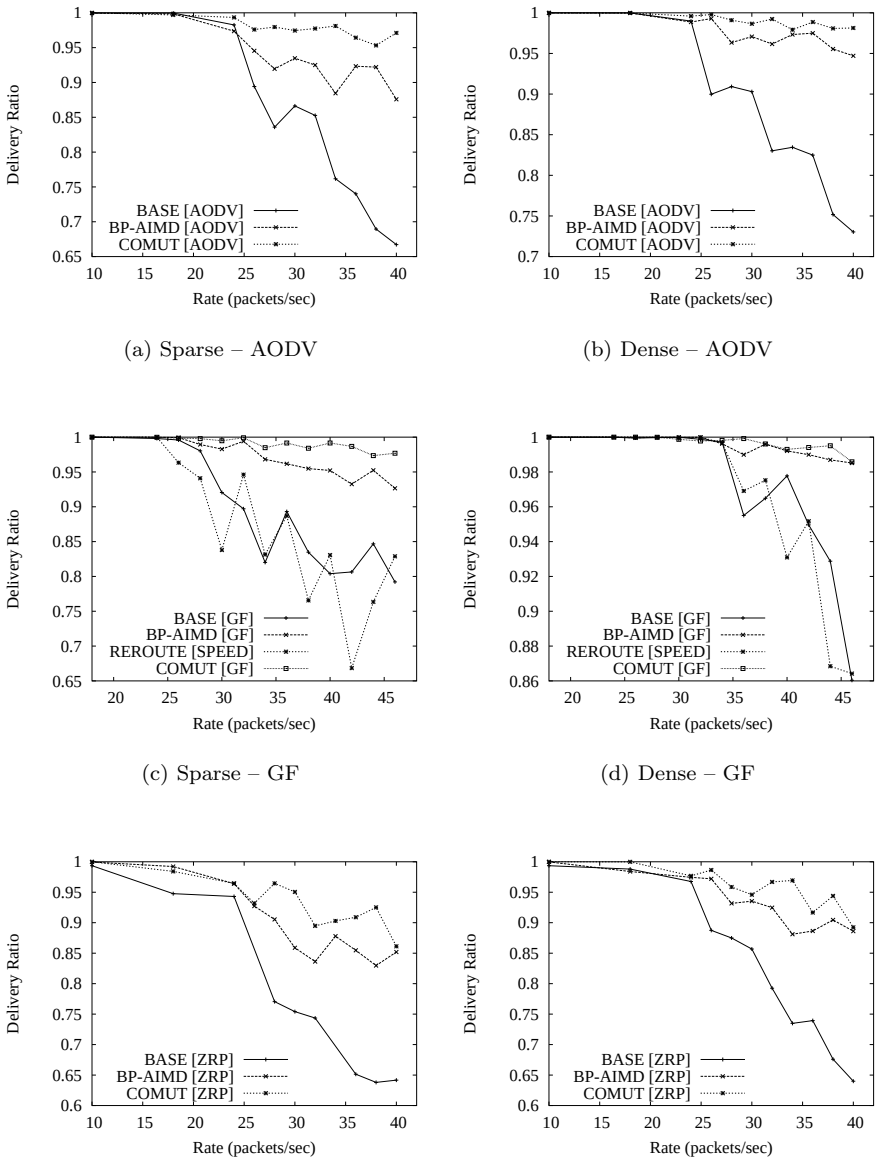
Fig. 11. γ Estimator vs Rate for all Clusters in Fig. 10 .

better than backpressure with BP–AIMD for all protocols and densities. Furthermore, we observe that the baseline technique (BASE) collapses at high frequencies of sending. Note that the delivery ratio for COMUT remains above 0.95 in most cases except when using ZRP in sparse networks. In sparse networks, the available routing paths are limited. Hence, almost any given path carries both the data and control messages from a multiplicity of sources and hence, have a tendency to become loaded. Due to the frequent and sometimes unnecessary ZRP updates, the problem is exacerbated and thus, COMUT (and BP–AIMD) control packets are delayed, resulting in delayed responses from the source. At higher densities, different routes are utilized to notify the sources and thus, the reaction is more timely. We will show later that although the delivery ratio is high, the bandwidth consumed for control messages for ZRP will come at the cost of reduced throughput, both with COMUT and BP–AIMD. However, despite the ZRP overhead, COMUT still maintains a delivery ratio of above 0.85.

Clearly, the delivery ratio results need to be considered in conjunction with the actual number of packets eventually received over the transmission period (throughput) to justify the use of rate control does not result in severe fidelity degradation. This is examined with a set of results on throughput later.

Although comparing different routing protocols is not our objective per se, we do notice that Geographical Forwarding allows for higher rates of packets to flow before the performance degrades due to overload. This is clear because, among all three routing protocols, it entails the least routing overhead (practically zero since it's stateless), while it finds the shortest path to the sink in terms of hops. We also observe that rerouting performs badly with respect to the delivery ratio especially in sparse networks. This is because alternate routes may not be available to a sensor when a forwarding neighbor broadcasts its inability to accept packets. Therefore, the sensor must resort to excessive drops of packets until an alternate route is found. In higher densities, more paths are available, a fact which improves the delivery ratio, although there still exist cases where alternate paths may be unavailable. When a sensor is overloaded, there is a high likelihood that the area around the sensor with its surrounding neighbors are also overloaded. Therefore, multiple backpressure rerouting messages may need to be sent until the congested area is bypassed, during which time, packets are dropped (to reduce the injection of traffic in the congested area).

Average Throughput: In Figure 13 we plot the number of useful packets received per flow over the entire simulation time (*i.e.* $\frac{Total\ Number\ of\ Packets\ Received\ at\ the\ Sink}{Total\ Time\ of\ Simulated\ Active\ Flows}$) with increasing rate. The error bars indicate the minimum and the maximum values observed for among all the performed experiments. The throughput reflects the perceived fidelity throughout the operation of the reporting process. The deviation between the sending rate and the average throughput indicates the resulting fidelity



(a) Sparse – AODV

(b) Dense – AODV

(c) Sparse – GF

(d) Dense – GF

(e) Sparse – ZRP

(f) Dense – ZRP

Fig. 12. Delivery ratio Vs Sending Rate.

loss. Clearly, due to bandwidth constraints, there exists a limit on the throughput that the network can support, given the specific message exchanges required by the routing protocol and the messages sent by the sensors themselves. This can be seen in the results of the simulations with BASE in Figures 13 (a)–(f) where the throughput seems to saturate and does not increase beyond some value. We observe that in most cases, the degradation of the throughput when utilizing rate control techniques is relatively small (given the resulting delivery ratios presented earlier) both with COMUT and BP–AIMD. When GF is used, the degradation with COMUT and BP-AIMD is more obvious. This is because GF, due to its stateless operation, achieves lower packet servicing delays than the other routing protocols. This allows for a higher volume of packets to be put in the outgoing queue by the sources. The new, high exogenous rate is higher than the stepwise increase in the rate with either rate control technique. Fine–tuning of the rate regulation intervals for COMUT and BP–AIMD based on the service time may provide better results in terms of throughput for these techniques. However, even with this degradation, the use of rate control is justified given the eventual lower delivery ratio of BASE and REROUTE discussed earlier. A final observation is that COMUT and BP–AIMD produce very similar throughputs.

Average Per-hop Delay: We now present the results obtained concerning the per-hop delay experienced using each method in Figure 14. The error bars indicate the minimum and the maximum values observed for among all the performed experiments. We measure the average per-hop delay, rather than the end-to-end delay, since each flow may be initiated at a different location in the network and might be closer to or further from the sink in terms of hops. We find that in almost all the cases, COMUT maintains very low per-hop delay (and consequently low end-to-end delay). Note here the advantage of COMUT over BP–AIMD: although COMUT and BP–AIMD both performed in a similar way in terms of delivery ratio and throughput, COMUT, in addition, manages to achieve lower delay. In Figures 14(c) and (d) we see that another technique, rerouting, manages to maintain very low delays comparable to COMUT. This is expected since (as the SPEED protocols requires) packets will be forwarded only if the delay requirements are met, therefore all received packets will be delivered in a timely manner. We remind, however, that rerouting experienced high losses, which was not the case for COMUT.

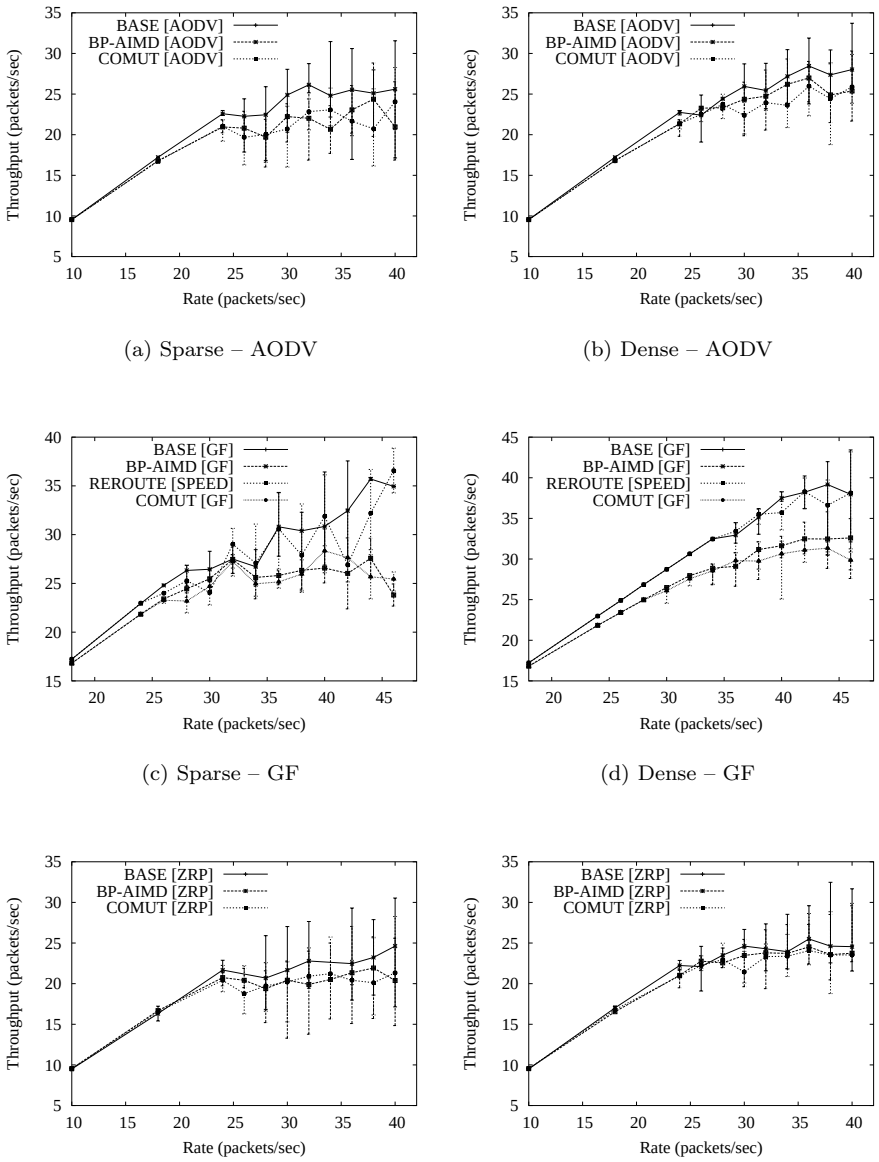
Energy Dissipation: In the next set of experiments we demonstrate the energy savings of COMUT and BP–AIMD over BASE, the non-congestion aware setting.

For the BASE scheme, we measure the wasteful transmissions by counting the number of MAC packet (re)transmissions $T_{Dropped}$[6] and the size of the data packet S_d (in bits); in particular, the total number of wasted bit transmissions $D = T_{Dropped} \cdot S_d$. This also reflects the wasted energy due to dropped packets. Measuring the wasted energy in terms of transmitted bits better captures the performance of the congestion control technique compared to measuring the residual energy. This is because the residual energy indicates the remaining energy with respect to the operation of the protocol but does not explicitly quantify the wasted energy due to packets drops.

To measure the energy wastage with backpressure, we consider, in addition to $T_{Dropped}$, T_b which refers to the number of hops a backpressure message was forwarded on its route to the source. Thus, if the size of the backpressure packets is S_b, we calculate $D_{BP} = T_{Dropped_{BP}} \cdot S_d + T_b \cdot S_b$.

Finally, with COMUT we must also consider the number of (re)transmissions or forwardings of control packets, T_c, of size S_c. This includes the number of sensor updates (one hop transmissions) plus the number of sentinel updates towards the upstream sentinel (multiple hops). Thus, with COMUT, the expended energy

[6] "Retransmissions" refers to all transmissions due to packet forwarding. A packet may be forwarded multiple hops and then suddenly dropped either due to queue overflow or MAC collision before reaching the sink. All the transmissions up to that point are, then, considered wasteful since they do not contribute to the effective throughput. Thus, the value of $T_{Dropped}$ incorporates the number of hops a packet was forwarded before it was dropped.



(a) Sparse – AODV

(b) Dense – AODV

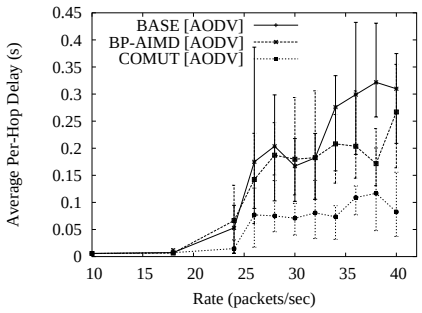
(c) Sparse – GF

(d) Dense – GF

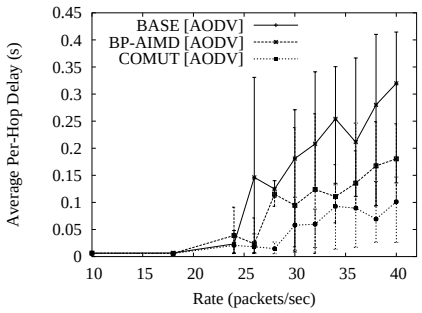
(e) Sparse – ZRP

(f) Dense – ZRP

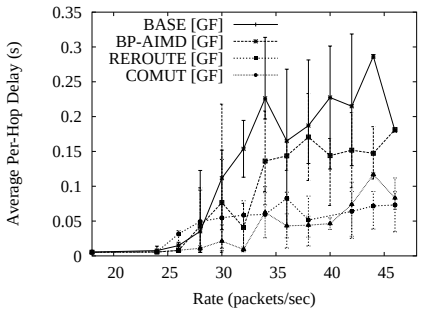
Fig. 13. Throughput Vs Sending Rate.



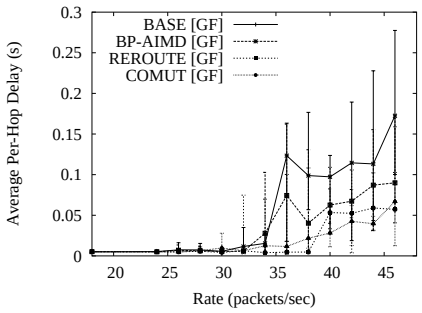
(a) Sparse – AODV



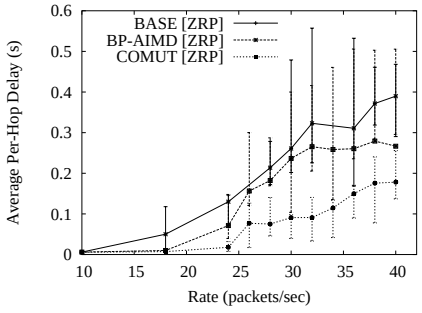
(b) Dense – AODV



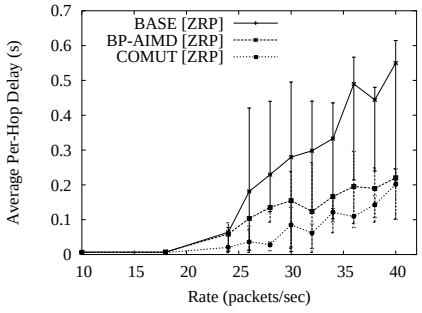
(c) Sparse – GF



(d) Dense – GF



(e) Sparse – ZRP



(f) Dense – ZRP

Fig. 14. Per Hop Delay Vs Sending Rate.

due to wasted transmissions and control traffic can be calculated as $D_{COMUT} = T_{Dropped_{COMUT}} \cdot S_d + T_c \cdot S_c$.

We represent the energy savings with Backpressure (R_{BP}) and and COMUT (R_{COMUT}) respectively, using the ratios: $R_{BP} = \frac{D-D_{BP}}{D_{BP}}$ and $R_{COMUT} = \frac{D-D_{COMUT}}{D_{COMUT}}$.

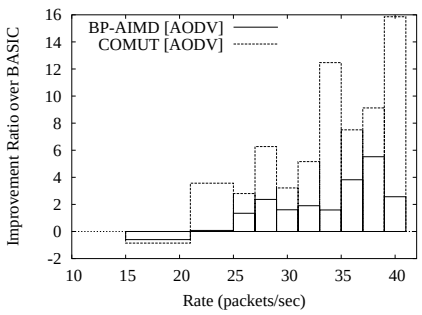
We plot R_{COMUT} and R_{BP} in Fig. 15 with increasing sending rate. We initially observe that for small transmission rates, the ratio is negative. Clearly, in the absence of congestion, the control overhead is wasteful and this is a direct artifact of this effect. This is the case with both COMUT and BP-AIMD. The proactive nature of COMUT may require the initiation of some sensor updates while, with BP-AIMD, a single dropped packet will cause forwarding of backpressure packets towards the source, possibly along multiple hops. 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 almost all cases, COMUT provides greater savings than BP-AIMD for all considered routing protocols.

One observation is that COMUT manages to keep the control messages, due to sensor and sentinel updates, low. We reiterate that updates both from sensors within a cluster as well as from sentinels across clusters, are only initiated after a threshold intensity level has been exceeded. This is the first factor that reduces the amount of traffic produced. Second, since COMUT is proactive, it anticipates the onset of congestion and quenches the overwhelming sources at an early stage. This has a positive reinforcing effect on COMUT since it reduces (or completely eliminates) further production of updates. In the case of the backpressure technique, messages are continuously produced as long as a queue overflows while congestion persists. In addition, such backpressure messages may be produced by many sensors within a congested area whereas, in contrast, COMUT incorporates all the sensor reports within a cluster into a single message.

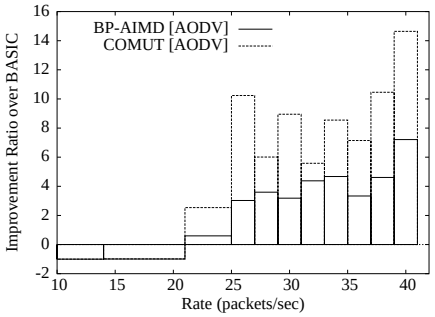
To summarize our observations thus far, COMUT combines the advantages of backpressure with respect to delivery ratio and throughput with the positive low delay property of rerouting. In addition, it manages to provide higher ratios of energy savings than backpressure when both are compared to the base setting. Finally, we our studies demonstrate that COMUT can operate effectively over multiple, different routing protocols.

Prioritization Support: We next provide our evaluation results with respect to flow prioritization. As stated, the goal of our policy is to allow for a higher volume of high priority traffic to be injected into the network. For our experiments, we introduce four flows into the network, first, two low priority flows and, then, two high priority flows. A flow is introduced every 1000 *ms*. The value of d_p is set to be 1 for high priority flows and 2 for the ones of low priority.

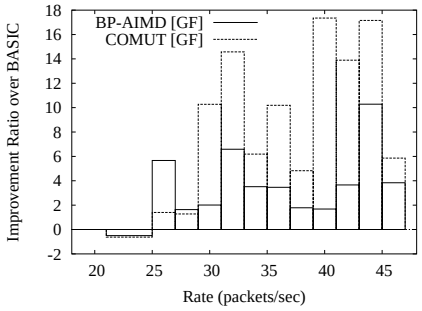
The resulting throughputs for each priority flow and for each routing protocol and network density are shown in Figure 16. We observe similar behaviors in almost all cases: when congestion is low, all flows are allowed to reach their maximum rate and maintain that rate throughout the completion of the experiment. However, as maximum requested rates increase, congestion occurs and lower priority flow sources are forced into dropping their rates. Therefore, we observe that the resulting throughput for the higher priority flows is closer to the requested maximum rate with COMUT, since bandwidth is still available for additional introduction of packets. More interestingly, in cases such as with low density networks (where congestion seems to have a more severe impact on the traffic (shown in Figures 16(a), (c) and (e)) the throughput of lower priority flows declines with increased maximum rates with COMUT. This allows for higher priority flows to maintain as high rates as possible. However, we do note that despite the fact that sources that send low priority packets drastically reduce their rate, this rate will never fall below the preset minimum rate. The delay and delivery ratios for both high and low priority traffic are not shown but are reflected in our previous experiments. As stated, COMUT does not sacrifice low priority packets or delay their transmissions. Instead, it stimulates proactive action at the sources and avoids the introduction



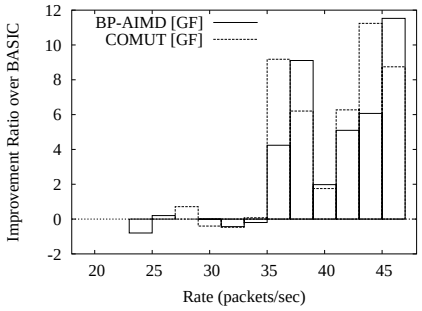
(a) Sparse – AODV



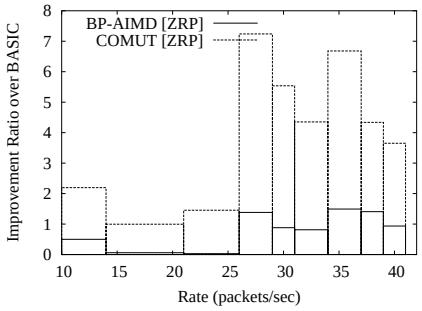
(b) Dense – AODV



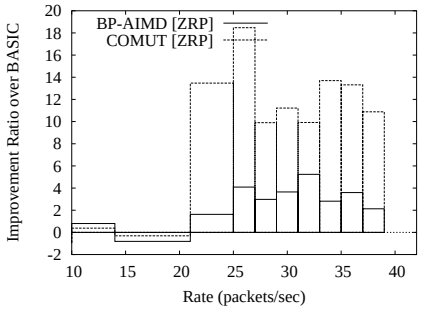
(c) Sparse – GF



(d) Dense – GF

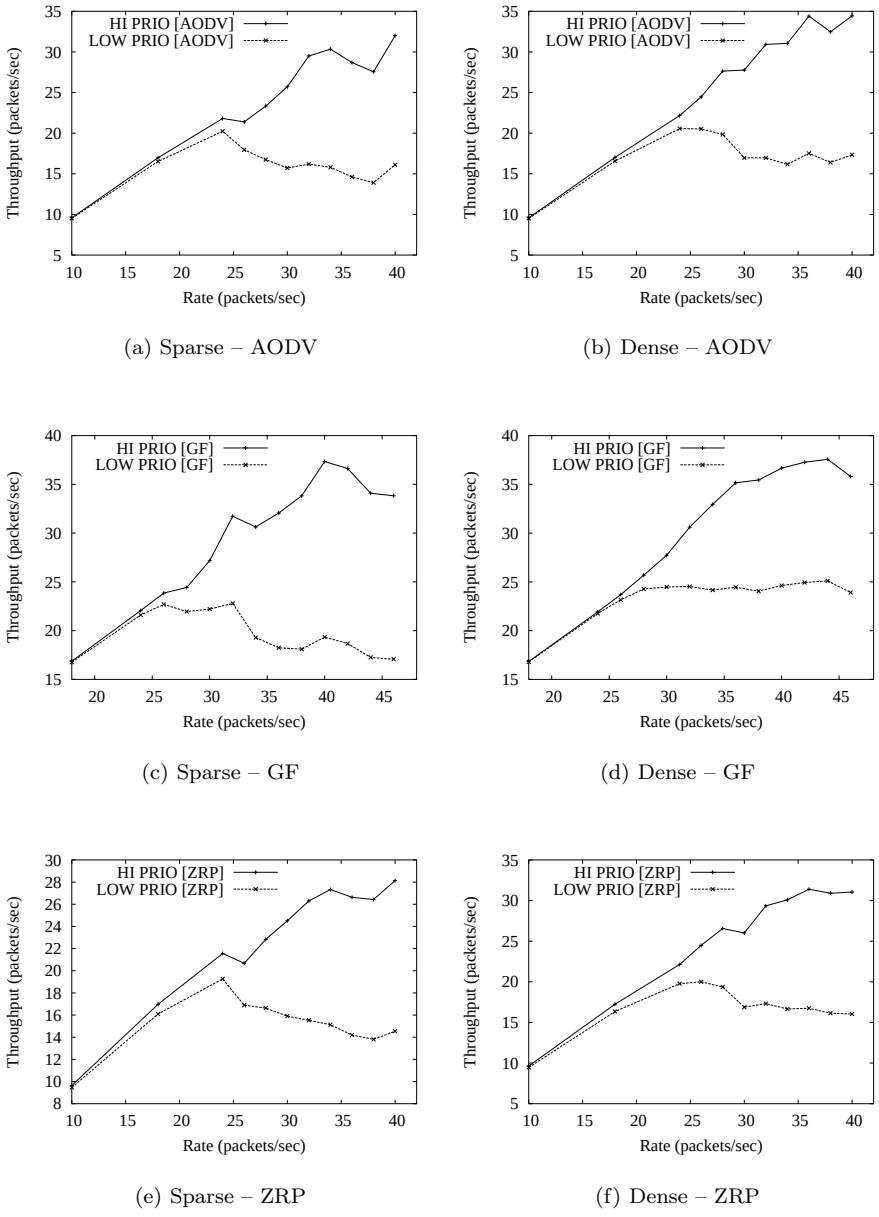


(e) Sparse – ZRP



(f) Dense – ZRP

Fig. 15. Energy Savings ratio over Base Technique.



(a) Sparse – AODV

(b) Dense – AODV

(c) Sparse – GF

(d) Dense – GF

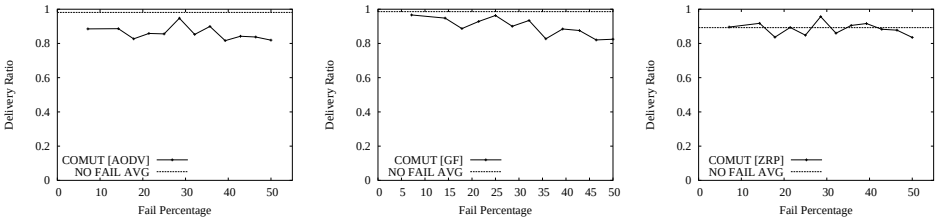
(e) Sparse – ZRP

(f) Dense – ZRP

Fig. 16. Average Rate (Throughput) assigned to High Vs Low Priority Flows.

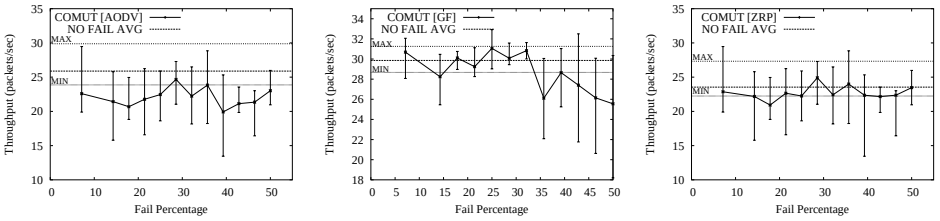
of excessive traffic. In this way, smooth network operation is maintained without energy-costly drops and low delays for all packets are achieved.

Sensor Failures: In this set of experiments, we illustrate the impact of sensor failures on COMUT. We have set up dense networks (140 sensors) so as to maintain reasonable connectivity even if a large number of sensors were to fail. During the simulation we distribute random failures of sensors, increasing the percentage of the total failed sensors at each step. We have reflected this percentage to selected



(a) Failures – AODV (b) Failures – GF (c) Failures – ZRP

Fig. 17. Sensor Failures: Delivery Ratio.



(a) Failures – AODV (b) Failures – GF (c) Failures – ZRP

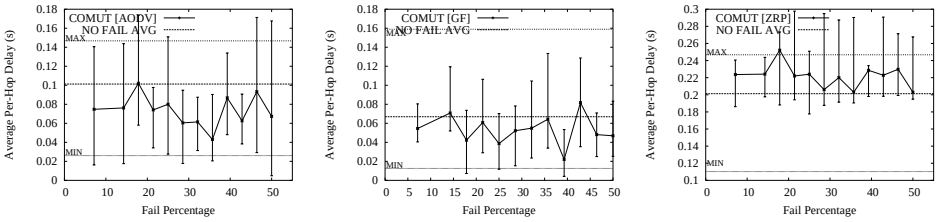
Fig. 18. Sensor Failures: Throughput.

sentinels as well as cluster sensors. We plot the obtained results for different routing protocols in Figures 17, 18 and 19.

Clearly, the route repair methods of various protocols affect COMUT. In Figure 17, we observe that the delivery ratio drops with AODV and GF. With AODV the reason is attributed to the fact that, while a repair is in progress, packets for which a route to the destination is not available, are dropped. With GF, even with high density networks routes may still become unavailable after a number of sensors die. In addition, with a high number of failed sensors, it is more probable that packets will need to be routed through the remaining sensors. However, even under these conditions, COMUT manages to maintain a delivery ratio of over 0.8 for both protocols. In contrast with AODV and GF, ZRP has an embedded repair mechanism periodically updating its neighborhood. For this reason, the delivery ratio remains near the average values as in the case without failures.

In Figure 18 we observe a reduction in the overall throughput. This is attributed to two factors. First, as above, there exist routing layer packet drops. Second, COMUT forces the sources to drop their rates due to increased traffic load especially with AODV and GF. The final observation is that ZRP throughput is least affected by the failures because of fewer drops, as discussed previously.

Finally, in Figure 19, we plot the observed average per hop delay. We find that the delay is slightly reduced when failures occur. We first remind that this delay is measured only for packets that are eventually received. With AODV and GF, COMUT forces the sources to reduce their sending rates. This results in overall reduction of the traffic load which also entails reduced delays. Furthermore, when paths fail, new paths are discovered and, therefore, it is possible that congestion ceases to persist in areas where multiple flows interfere with each other.



(a) Failures – AODV (b) Failures – GF (c) Failures – ZRP

Fig. 19. Sensor Failures: Average Per-hop Delay.

7.3 Incorporating More Realistic Channel Effects

With the relatively simple path-loss channel model, we illustrated the functionality of COMUT over multiple routing protocols. We now extend our tests to include a different channel model to study COMUT's functionality in a more realistic setting.

Fading Channel Model: It has been shown that a path-loss exponent component and log-normal fading component together model very accurately, cases of short range line-of-sight communications. As shown in [Krishnamachari 2005], this model is fairly accurate in depicting the channel observed in sensor networks. The channel effects may be captured by the following equations:

$$P_r(d) = P_t - PL(d) \tag{4}$$

$$PL(d) = PL(d_0) + 10\eta log(\frac{d}{d_0}) + X_\sigma \tag{5}$$

In the above equations, given a receiver-transmitter distance of d, $P_r(d)$ is the received power in dB, P_t the transmit power, $PL(d)$ is the path loss in dB, d_0 is a reference distance and X_σ is a Gaussian random variable (in dB) with variance σ and mean equal to 0. This model is able to capture the link packet reception statistics observed empirically as shown in [Zuniga and Krishnamachari 2004] and [Krishnamachari 2005].

Considering the Effects of Probabilistic Reception: In sensor networks, the effects of a fading channel (especially for routing) were studied and quite effective solutions (such as *blacklisting* [Gnawali et al. 2004]) have been proposed in the literature to complement higher layer control schemes such as COMUT. In our work, we also utilize MAC layer retransmission-based mechanism (ARQ). However, we need to further consider the specific requirements of COMUT with respect to these effects. More precisely, we need to consider the realistic impact of *probabilistic* packet reception. Recall that earlier we assumed reliable packet delivery within a prespecified range and no delivery outside this range. Under that model we were able to ensure that sensor nodes within the transmission range would definitely establish a link with the sentinel and would be able to transmit updates. Under the probabilistic model this assumption seizes to hold. To address this problem, we must ensure that communication among sentinels and its cluster members can be enabled *with high probability*. Clearly, link establishment is dependent on the model parameters. In order to decouple the dependence on the environment parameters[7], we can alternatively require that nodes are dense enough to ensure link establishment as well as integrate *link quality measurements* in the sentinel selection phase. We describe these two choices next.

[7]We have selected the array of channel parameters based on the comparisons between analytical and empirical studies in [Krishnamachari 2005].

Network Density Requirement for Sensor-Sentinel Link Establishment: Empirical studies in [Krishnamachari 2005] and [Zuniga and Krishnamachari 2004] have demonstrated a separation of sensor communications into three distinct regions with respect to the transmitter-receiver distance: a *connected region*, the *transitional region* and a *disconnected region*. In the connected region the Signal-to-Noise (SNR) ratio is such that connectivity exists with very high probability; the *transitional region* is the region within which the link quality shows significant variations and finally, in the disconnected region the received power is consistently below the minimum connectivity threshold. Our goal, then, is to ensure that cluster nodes and sentinels are within the connected region.

To achieve the above we first turn to some specific theoretical results. In [Bettstetter and Hartmann 2003] and [Stuedi et al. 2005] the authors produced theoretical results on the connectivity requirements in shadow fading environments. We utilize the results from the former to first extract the distance which correspond to the beginning of the transitional region. We will refer to this distance as d_{thres}. Then we will utilize a simple probabilistic analysis to produce an estimate on the number of sensors required to achieve connectivity. In [Bettstetter and Hartmann 2003] the link probability was found to be:

$$Pr(link) = \frac{1}{2} - \frac{1}{2}erf(\frac{10\eta}{\sqrt{2}\sigma}\log\frac{d}{r_0})$$

where $P(link)$ is the probability that a link exists between two nodes, and r_0 is the normalization distance, i.e. the maximum distance at which a link can be established with no shadow fading and is effectively the distance when setting σ equal to 0. Using $\sigma = 2$, $\eta = 2$, r_0 =25m, and link probability 0.99, the beginning of the transitional region, d_{thres}, is at approximately 15m (see also [Bettstetter and Hartmann 2003]. In fact, this result matches closely the empirical studies reported in [Krishnamachari 2005]. We would now like to estimate, given any node in a geographical square region with area A, the probability that is disconnected, or, in other words, the probability that no other node is within its d_{thres} range. For N nodes, this probability is $\Pr(disconnect) = (1 - \frac{\pi d_{thres}^2}{A})^N$. Thus, if we require a disconnection probability almost zero, i.e. $\Pr(disconnect) < e$ then the number of nodes in A is $N \geq \frac{\log(e)}{\log(1-\frac{\pi d_{thres}^2}{A})}$. For example, in our tests we used the parameters above (*i.e.* an area of 100x100m^2, $d_{thres} = 15m$), and a disconnection probability of $e = 0.0001$, which yields an estimate on the node density to about 126 nodes.

Link Quality Statistics – Routing and Sentinel Selection: We note that the node density requirement does not guarantee that multi-hop paths chosen by some routing technique will be reliable. Clearly, however, a routing protocol that chooses paths based on the link quality would perform better [Rangwala et al. 2006]. We use Geographical Forwarding as the routing layer, enhanced with selecting the node with the higher link-quality and distance-to-destination ratio as its next hop (as in MultiHop implemented in TinyOS [TinyOS]). For this technique we measure link state statistics (received power) from the physical layer as in [Woo and Culler 2001]. In addition, during the sentinel selection phase, a node does not ignore additional sentinel advertisement message but instead chooses the sensor node with the strongest signal as its sentinel.

Results and Discussion: In a lossy channel, packets that cannot be received correctly must be retransmitted contributing to the service delays and energy expenditure. *This artifact stresses the need for effective rate control especially in stringent-resource sensor networks.* With respect to this observation, early congestion reaction is utilized effectively by COMUT as illustrated next. Increased packet retransmission at higher packet generation rates further increase the service delays which, in tern, affect the load of the node. COMUT captures this situation and proactively alleviates its effects. In Fig. 20 we plot the goodput (Total number of unique packets received at the sink throughout the simulation time per

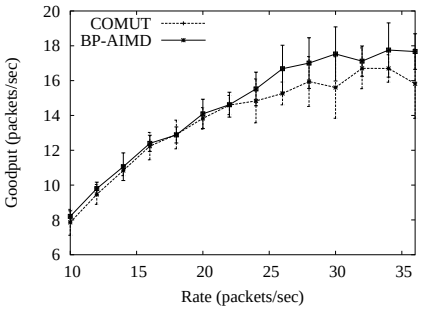


Fig. 20. Goodput Vs Rate.

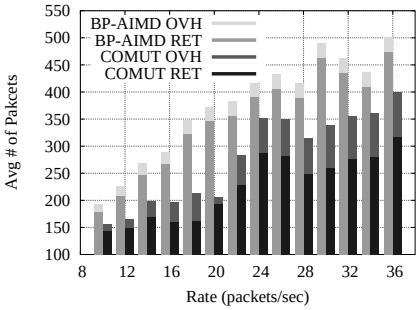


Fig. 21. Retransmission and Control Packet Overhead Vs Rate.

flow). In Fig. 21 we show the overhead in terms of packets broken down into retransmissions/duplicates (RET) and control overhead (OVH). Overall, we find that compared to BP-AIMD, an average reduction of about 7% in goodbput yields an average of about 20% reduction in overhead transmissions.

8. RELATED WORK

In this section, we discuss the related work on congestion control and service differentiation in sensor networks.

CODA [Wan et al. 2003] is a congestion detection and avoidance scheme for sensor networks that combines local backpressure techniques [Wang et al. 2005] and sink-to-sensors notifications but is not specifically concerned with different classes of packets. Unlike COMUT, where the rate is regulated in a proactive manner in order to prevent losses of high priority data, CODA uses a reactive technique.

A more recent work [Hull et al. 2004] proposes techniques that are similar to those in CODA while providing detailed cost metrics to evaluate the performance of sensor networks under congestion in the context of realistic workloads. Protocols such as CODA may extend their *reactive* backpressure techniques used in order to stall low priority packets in the presence of high priority packets. However, these schemes do not prevent congestion, thus, are not resistant to sensitive drops of high priority data.

In [Ee and Bajcsy 2004], 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 (effective capacity on a link) among the children nodes. A more detailed fairness-aware technique as well as analysis on the effective selection of the AIMD parameters was also presented in [Rangwala et al. 2006]. Our work differs in that we target sensor coordination at the cluster level, which considers interactions among multiple, distinct flows. ESRT [Sankarasubramaniam et al. 2003] is another transport layer solution which regulates the sensors' sending rates so that reliability (with respect to the number of packets received) is achieved while avoiding congestion. ESRT, however, is better suited for sensor networks geared towards monitoring applications that report values periodically.

There has been some work on priority support in ad hoc wireless networks. Yang and Kravets [Yang and Kravets 2004] propose an admission control mechanism for real time and best effort traffic in a wireless ad hoc network. Unlike with COMUT, in [Yang and Kravets 2004], a real-time flow might not be admitted because it may degrade the throughput of other pre-existing real-time flows. In our work, two flows of equal priority are allowed to enter the network; this is because important packets must reach the sink even if their assigned bandwidth might eventually be shared. QoS scheduling for bursty flows (as in [Zhu and Cao 2004]) allows high priority flows to claim bandwidth assigned to low priority flows. However this requires the use of TDMA which is difficult to enforce in a sensor network of randomly deployed

sensors without a central arbiter.

MAC layer prioritization schemes such as [Yang and Vaidya 2002] provide extensions to the IEEE 802.11 protocol to allow for higher priority packets to be sent out from a local queue before low priority packets [Yang and Vaidya 2002; Lu et al. 2002]. One might provide the higher priority packets with lower back-off times that would allow them to access the channel with higher probabilities [Yang and Vaidya 2002], [Lu et al. 2002]. However, even with these schemes, due to interference range effects, low priority packets contend with those of high priority. Furthermore, loss of control packets could still occur causing route failures and other associated effects. While MAC layer prioritization can effectively supplement rate control, it cannot replace rate control to alleviate congestion.

SPEED is a routing protocol that uses the *speed* (defined next) between neighboring nodes along the path as the priority metric to distinguish flows [He et al. 2003]. Speed is defined as the actual speed with which packets can be transported from one hop to another *i.e.,* the ratio of the distance to the delay. RAP [Lu et al. 2002] utilizes a similar technique to provide soft real time guarantees by locally considering the *velocity* at a node. These protocols do not consider the interactions of multiple flows with statically assigned priorities. Furthermore, they resort to backpressure (as discussed earlier) when congestion occurs; it is explicitly stated that a certain percentage of packets may need to be dropped with these techniques to reduce the traffic injected into the congested area. Packet drops are not desirable; they lead to energy wastage and could potentially result in the loss of important packets. Furthermore, Geographic Forwarding – the base routing scheme assumed with these protocols – requires location information that could be potentially expensive in sensor networks.

9. 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 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 framework incorporates (i) a proactive initialization phase that provides indications of the flows' paths, (ii) periodic intra-cluster traffic intensity estimation, (iii) periodic inter-sentinel cooperation, and (iv) controlled rate at the sources. Our simulations show that COMUT is highly successful in abating congestion, in reducing wasteful packet drops in the network and providing high throughput to important flows. We also demonstrate that COMUT is able to operate successfully over multiple underlying routing infrastructures and fading channel conditions while being perceptive and responsive to sensor failures.

REFERENCES

Amis, A., Prakash, R., Huynh, D., and Vuong, T. 2000. Max-min D-Cluster formation in wireless ad hoc networks. In *Proceedings of the 2000 IEEE Computer and Communications Societies Conference on Computer Communications (INFOCOM)*. Tel Aviv, Israel, 32–41.

Amis, A. D. and Prakash, R. 2000. Load-balancing clusters in wireless ad hoc networks. In *Proceedings of the 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology (ASSET)*. IEEE Computer Society, Washington, DC, 25.

Bandyopadhyay, S. and Coyle, E. 2003. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In *Proc. of the 22nd IEEE INFOCOM*. San Francisco, CA, 1713–1723.

Basagni, S. 1999. Distributed clustering for ad hoc networks. In *ISPAN 99: Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks*. Fremantle, Australia, 310.

Bechler, M., Hof, H.-J., Kraft, D., Pählke, F., and Wolf, L. 2004. A cluster-based security architecture for ad hoc networks. In *Proc. of the 23rd IEEE INFOCOM*. Hong Kong, 2393–2403.

Bertsekas, D. P. and Gallager, R. 1992. *Data Networks*, 2nd ed. Prentice Hall.

Bettstetter, C. and Hartmann, C. 2003. Connectivity of wireless multihop networks in a shadow fading environment. In *MSWiM , 2003*. San Diego, CA, 28–32.

Ee, C. T. and Bajcsy, R. 2004. Congestion control and fairness for many-to-one routing in sensor networks. In *SenSys*. Baltimore, MD, 148–161.

Elson, J. and Estrin, D. 2001. Time synchronization for wireless sensor networks. In *Proc. of the IEEE International Parallel & Distributed Processing Symposium IPDPS*. San Francisco, CA, 186.

Gnawali, O., Yarvis, M., Heidemann, J., and Govindan, R. 2004. Interaction of retransmission, blacklisting, and routing metrics for reliability in sensor network routing. In *Proc. IEEE SECON*. Santa Clara, CA, 34–43.

Haas, Z. J. and Pearlman, M. R. 1998. The zone routing protocol. Internet Draft.

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

He, T., Stankovic, J. A., Lu, C., and Abdelzaher, T. F. 2003. SPEED: A stateless protocol for real-time communication in sensor networks. In *Proc. of the 23rd Int'l Conf. on Distributed Computing Systems*. Providence, RI, 45–55.

Heinzelman, W. R., Chandrakasan, A., and Balakrishnan, H. 2000. Energy-efficient communication protocol for wireless microsensor networks. In *Proc. of the 33rd Hawaii Int'l Conf. on System Sciences-Vol.e 8*. Hawaii, HI, 8020.

Hull, B., Jamieson, K., and Balakrishnan, H. 2004. Mitigating congestion in wireless sensor networks. In *SenSys*. Baltimore, MD, 134–147.

Karenos, K., Kalogeraki, V., and Krishnamurthy, S. V. 2005. Cluster-based congestion control for supporting multiple classes of traffic in sensor networks. In *Proc. of the 2nd IEEE Workshop on Embedded Networked Sensor Systems*. Sydney, Australia, 107–114.

Krishna, P., Vaidya, N. H., Chatterjee, M., and Pradhan, D. K. 1997. A cluster-based approach for routing in dynamic networks. *SIGCOMM Comput. Commun. Rev. 27,* 2, 49–64.

Krishnamachari, B. 2005. *Networking Wireless Sensors*. Cambridge University Press, New York, NY.

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

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

Lin, C. R. and Gerla, M. 1997. Adaptive clustering for mobile wireless networks. *IEEE Journal of Selected Areas in Communications 15,* 7, 1265–1275.

Lu, C., Blum, B. M., Abdelzaher, T. F., Stankovic, J. A., and He, T. 2002. RAP: A real-time communication architecture for large-scale wireless sensor networks. In *IEEE Real Time Technology and Applications Symposium*. San Jose, CA, 55–66.

Mhatre, V. and Rosenberg, C. 2004. Design guidelines for wireless sensor networks: Communication, clustering and aggregation. *Ad Hoc Networks Journal 2,* 1, 45–63.

motes. Crossbow mica motes. Available at www.xbow.com.

ns. Network simulator NS–2. Available at www.isi.edu/nsnam/ns.

Perkins, C. E. and Royer, E. 1999. Ad-hoc on demand distance vector routing. In *Proc. IEEE Workshop on Mobile Computing Systems and Applications*. New Orleans, LA, 90–100.

Rangwala, S., Gummadi, R., Govindan, R., and Psounis, K. 2006. Interference-aware fair rate control in wireless sensor networks. In *ACM SIGCOMM*. Pisa, Italy, 63–74.

Sankarasubramaniam, Y., Akan, B., and Akyildiz, I. F. 2003. Esrt: event-to-sink reliable transport in wireless sensor networks. In *Proc. of the ACM International Symposium on Mobile Ad hoc Networking & Computing*. 177–188.

Stuedi, P., Chinellato, O., and Alonso, G. 2005. Connectivity in the presence of shadowing in 802.11 ad hoc networks. In *WCNC*. New Orleans, LA, 2225–2230.

TinyOS. Operating environment for embedded networked sensors. Available at www.tinyos.net.

Wan, C.-Y., Campbell, A. T., and Krishnamurthy, L. 2002. Psfq: a reliable transport protocol for wireless sensor networks. In *Proceedings of the 1st ACM international workshop on Wireless Sensor Networks and Applications (WSNA)*. ACM Press, New York, NY, 1–11.

Wan, C.-Y., Eisenman, S. B., and Campbell, A. T. 2003. CODA: congestion detection and avoidance in sensor networks. In *SenSys*. Los Angeles, CA, 266–279.

Wang, C., Sohraby, K., and Li, B. 2005. Sentcp: A hop-by-hop congestion control protocol for wireless sensor networks. In *IEEE INFOCOM 2005 (Poster Paper)*. Miami, FL, 107–114.

Woo, A. and Culler, D. E. 2001. A transmission control scheme for media access in sensor networks. In *Proc. of MobiCom*. Rome, Italy, 221–235.

Woo, A., Tong, T., and Culler, D. E. 2003. Taming the underlying challenges of reliable multihop routing in sensor networks. In *SenSys*. Los Angeles, CA, 14–27.

Wu, J. 2002. Dominating-set-based routing in ad hoc wireless networks. *Handbook of wireless networks and mobile computing*, 425–450.

YANG, X. AND VAIDYA, N. H. 2002. Priority scheduling in wireless ad hoc networks. In *Proc. of the 6th ACM Int'l Symposium on Mobile Ad Hoc Networking and Computing*. Lausanne, Switzerland, 71–79.

YANG, Y. AND KRAVETS, R. 2004. Throughput guarantees for multi-priority traffic in ad hoc networks. In *Proc. of the IEEE Int'l Conf. on Mobile Ad-hoc and Sensor Systems*. Fort Lauderdale, FL, 379–388.

YOUNIS, O. AND FAHMY, S. 2004. Distributed clustering in ad-hoc sensor networks: A hybrid energy-efficient approach. In *Proc. of IEEE INFOCOM*. Hong Kong, –640.

ZHU, H. AND CAO, G. 2004. On improving service differentiation under bursty data traffic in wireless networks. In *Proc. IEEE INFOCOM*. Hong Kong, 871– 881.

ZUNIGA, M. AND KRISHNAMACHARI, B. 2004. Analysing the transitional region in low-power wireless links. In *Proc. IEEE SECON*. Santa Clara, CA, 517– 526.