

A Rate Control Framework for Supporting Multiple Classes of Traffic in Sensor Networks*

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

Abstract

Wireless sensor network applications typically integrate, within the same network, a variety of sensing devices including those for imaging, sound and temperature. In these settings, multiple flows of packets with different requirements in terms of transmission rates, bandwidth and jitter demands may be initiated towards the sink. Uncontrolled introduction of traffic from sources can cause network overload in areas of the network where the paths of the different flows interfere with each other. Such interference effects may result in congestion which leads to high packet loss and excessive delays. In this paper, we present CoBRA, a framework which incorporates distributed, cluster-based mechanisms to address the problem of congestion by enforcing rate control, for supporting multiple classes of traffic in sensor networks. Towards this goal, CoBRA periodically estimates the collective traffic load and, based on the current conditions, allocates and adjusts rates to sources on per-cluster bases. While doing so, CoBRA takes into consideration interference effects and rate requirements of concurrent flows. We have applied two different rate allocation policies using our framework and, through extensive simulation results we demonstrate its feasibility, effectiveness and performance advantages over traditional approaches.

1 Introduction

Advances in wireless sensor network communication protocols and low-power hardware devices have promoted the proliferation of a variety of sensor network applications, including environmental and habitat monitoring, agriculture, surveillance and emergency response, that integrate, within the same network, a multiplicity of sensor types such as imaging, temperature and sound. With these different classes of applications, the types of sensors used may initiate multiple flows that have diverse requirements in terms

of transmission rate, delay and throughput, towards the sink. For example, multimedia applications have bandwidth, delay and jitter demands, while the demand of an emergency response application is the availability of the network itself.

The initiation of multiple types of simultaneous flows without controlling their sending rates is likely to cause congestion, first, due to network overload and second, due to collisions of packets from simultaneous flows traversing interfering paths from the plurality of sources to the sink. Congestion can cause enormous delays as well as packet drops. Furthermore, dropped packets that are forwarded over multiple hops but never delivered at the sink, result in energy wastage. Thus, to support multiple classes of applications in the sensor network, our goal is to achieve a more deterministic network behavior so that flows are delivered based on the pre-specified requirements of the application classes and network resources are better utilized.

The characteristics of sensor networks such as limited availability of resources, interference-coupled paths and the lack of a central coordinator, make the problem more challenging. First, in event-based sensor environments, the number of sensors is typically large while the number of sinks is relatively small. When multiple events occur concurrently, they can create several flows that could largely interfere with each other. Even with small numbers of simultaneous flows, it may be impossible to bypass congested areas; thus, serious delays and packet losses may be caused due to contention and medium saturation. Hence, one important question is how to estimate the maximum rate that can be allocated to a new flow while it is contending with other concurrent *interfering* flows. For example, the addition of a new flow can potentially decrease the throughput and increase the delay of existing flows. Although there exist theoretical limits that estimate the maximum sustainable network capacity [14], doing such estimations dynamically and within the localized scope of the sensor nodes is hard.

A second challenge is how to detect congestion and estimate the current *traffic intensity*, an indicator of the congestion level, in the sensor network. With local, per-sensor

*This research has been supported by NSF Award 0330481.

estimates only, congested flows across the network may not be identified. Single node estimates, such as monitoring the sensor's queue size, may also be misleading. For example, a full queue, might indicate high queue utilization and not necessarily suggest congestion since the queues of the neighbors may be empty. In addition, single sensor estimates may not effectively capture the effects of multiple flow interference. One approach is to have the multiple sensor nodes exchange traffic information through control packets. This would allow the nodes to obtain a more accurate representation of the flow interactions. However, unnecessary propagations are energy costly and may increase the network traffic, thus, further contributing to congestion.

Current congestion control or admission control approaches are not adequate to address these problems. Congestion control approaches that have been proposed [4, 10, 19], adjust the network load using rate control techniques. However, these techniques are reactive in nature and are implemented only after congestion has already occurred and, thus, may not be able to avoid excessive packet losses. Furthermore, these approaches fail to take into consideration the different requirements of the individual classes of flows. On the other hand, bandwidth allocation and prioritization techniques for wireless ad hoc networks [13, 17, 22] do not consider congestion and its effects, while end-to-end resource reservation mechanisms [3, 5, 12] may lack timeliness and reactivity.

In this paper we propose CoBRA, a (**CO**ngestion-**B**ased **R**ate Allocation) framework that implements rate allocation and control to address the problem of congestion and provide support for multiple classes of flows in sensor networks. CoBRA provides a distributed congestion estimation technique that *proactively* monitors the network traffic on a *per-cluster* basis. Using an approximate traffic modeling approach for estimating the network's traffic intensity, CoBRA determines whether a new flow can be added based on the number of active flows and the maximum aggregate rate that can be allocated to the flows. The question we want to answer is, "*Given the current network conditions, is it possible to add a new flow in the network at a specific rate, without producing congestion or severely degrading other flow rates? If this rate cannot be accepted, what rate can the network support?*" In our framework, regulation of sensor rates occurs at the sources based on the classes of applications and the congestion state en route the sink. CoBRA implements two algorithms for rate allocation: admission control and proportional rate allocation. The distinguishing characteristic of our techniques is that they allow for timely localized decision making and management of the level of congestion. Using extensive simulations, we show that our approach is more responsive than traditional rate control techniques and manages to effectively and fairly allocate rates to flows belonging to multiple classes of traffic.

2 Network Model

We consider applications deployed in *event-based*, multi-hop sensor networks. An example of such a network would be one constructed during a *disaster recovery rescue mission*: Sensors are placed in the disaster area and are programmed to report events to a sink. Examples of such events are capturing and sending images when movement is detected, sending alarm messages in the presence of fire or, generating sensor status reports (such as low battery level or low signal power). We describe the main design features of our approach:

Support for Multiple Traffic Classes: Events sensed by different sensing devices may have different requirements with respect to transmission rates and throughput at the sink. For example, multimedia images will require higher transmission rates and higher throughput than alarm reports. Hence, our goal is to allocate rates to each initiating flow based on its application class. This is important, because, if rates are allocated unfairly to the different classes, congestion might cause dropped packets as well as arbitrary delays. In these cases, it may be more important for the sink to receive low rates from all types of sensors in the geographical region of interest rather than a high rate from a single sensor.

Proactive Rate Control: In disaster recovery missions, events are likely to be detected at random times and at random places in the sensor network, thereby dynamically changing the interference patterns among flows. This characteristic hinders our ability to predict the network behavior and effectively identify points of congestion. In addition, the presence of multiple classes of traffic makes the problem even more difficult. Our goal is to *proactively* monitor the network and identify or predict the onset of congestion. A propagation scheme is also required for the information to be quickly exchanged between the sensors and to notify the sources that route packets through the congested area to reduce their sending rates. Such proactive rate control allows us to anticipate unpredictable injection of flows and avoid dropping critical packets upon congestion.

Timely Event Reporting: Events must be reported in *real-time* after they occur. However, in a congested network, delays dramatically increase. More specifically, delays are attributed to processing, transmission and queuing of the events as they propagate across multiple sensors in the network. Among these, queuing delay and transmission delay are the predominant delay factors as the network load increases. Delays due to contention can be considered part of the queuing/processing delay since, as contention grows, more packets build up in the sensor queue increasing their service times. Thus, effectively managing the traffic load can lead to considerable reduction in delays. Due to the unpredictability in the initiation of flows and the diverse re-

quirements of the flows with respect to their sending rates, congestion control becomes challenging.

Network Parameters: We define a number of measures to evaluate the performance of our approach. To represent the current network load conditions we define the local load observed by a sensor as ρ and we use the *Traffic Intensity*, denoted as γ , to represent the collective network load. More formally, we define γ as the probability of at least one packet being present at any sensor queue in the network. The *Traffic Intensity Threshold* γ_{thres} defines the desired network load that does not underutilize the network nor causes congestion. This allows us to identify whether congestion has occurred. Calculation of γ and selection of γ_{thres} are provided in Section 3.2.

A *flow* i is defined as a tuple $F_i\{s_i, r_i, a_i, c_i\}$, where s_i is an identifier that specifies the source sensor and the particular flow itself, and r_i and a_i are the requested and allocated rates pertaining to this flow. Multiple flows may be initiated from the same source sensors; each flow has its own r_i and a_i which are based on the current Traffic Intensity measurement and may vary with time. Furthermore, a flow may request multiple times to be added to the network (for example, when other flows complete and there are available resources). Finally, c_i represents the *Flow Class*. Classes for flows are application-specific. A class determines the rate that should be allocated to a new flow based on the Rate Allocation Policy.

A *Rate Allocation Policy* is defined as a function A which takes as input, the requested rate and the flow class parameters and returns the allocated rate for each flow. In order to allocate a rate to a new flow, A must be provided with an estimate of the maximum aggregate rate R_{est} that can be supported in the network without causing γ_{thres} to be exceeded. The *Aggregate Estimated Maximum Rate*, R_{est} , is defined as the rate that, when allocated to all interfering flows, the Traffic Intensity becomes equal to γ_{thres} .

One important question is how to estimate R_{est} . Given a specific rate, we would like to calculate the expected value for γ . Modeling the behavior of γ with changing rate allows us to predict the value of R_{est} . More specifically, given the aggregate rate allocated to all the flows active along paths that interfere with each other (R_{tot}) and the Traffic Intensity γ resulting from this rate, we can use the model to identify the additional rate (positive or negative) at which γ_{thres} will be achieved. Note that R_{tot} is only concerned with a particular collection of flows that traverse interfering paths. We provide an approximate model for capturing the behavior of the traffic intensity in Section 3.3.

3 CoBRA Framework

Our proposed **CO**ngestion-**B**ased **R**ate Allocation (**CoBRA**) framework is designed to handle congestion by pro-

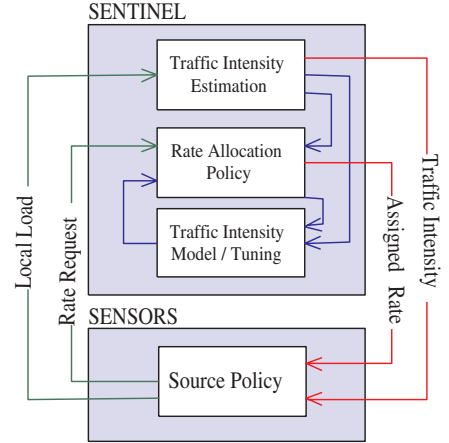


Figure 1. CoBRA Framework Components.

viding fast feedback with respect to the current congestion level and to support localized decision making for allocating rates to flows of multiple classes. Decisions are taken based on the congestion level of the network. CoBRA consists of the following components: (i) a *Cluster-based Network Structure* to support decentralized decision making, (ii) a *Distributed Traffic Intensity Estimation* process that uses a (iii) *Traffic Intensity Model* and (iv) *Rate Allocation Policies* employed at the clusters and the sources. The interactions between the components of the CoBRA framework are illustrated in Figure 1.

3.1 Cluster-based Network Structure

Our framework's operation is driven by the underlying architectural constructs. We adopt a technique proposed previously for grouping sensors in order to provide collective and distributed functionality [6, 11]. This technique is based on organizing the network into *clusters*. Each cluster is governed by an appointed *sentinel* sensor, *i.e.*, a cluster-head. Sensors locally compute their traffic load and send it to the sentinel via a *single-hop* broadcast. The sentinel then processes the reported values to produce a collective cluster estimation of the Traffic Intensity.

We have chosen a cluster-based solution due to its multiple benefits. Clustering offers scalability and allows for highly accurate estimations since a collection of sensors can better capture interactions between multiple flows. In addition, the selection of a single sensor as a representative for the whole cluster allows for the aggregation of updates to be propagated to the sources. Furthermore, cluster-based solutions allow us to achieve relatively low overhead since only clusterhead sensors are required to exchange control information. The overhead is justified by the major energy savings achieved due to the reduction in wasteful packet drops

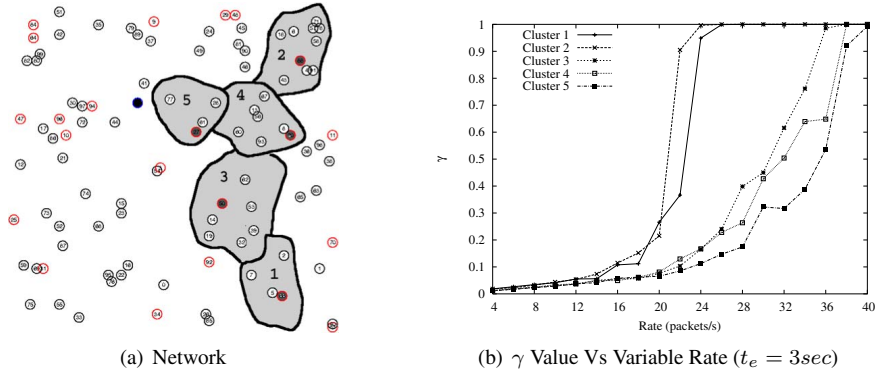


Figure 2. The Simulated Network. Each curve in (b) corresponds to a cluster in (a).

in turn leading to squandered energy.

The cluster construction is completed in two steps: The first step is the *election phase* in which, with some probability P , sensors broadcast a sentinel announcement message after the lapse of a random timer. While waiting for the timer to expire, if a sentinel announcement message is received, the sensor will simply join the sender's cluster and stop. Else, the probability P increases. If after the increase of P , the sensor has still not become a sentinel, the timer is reset. Eventually, after a small number of iterations, each sensor either becomes a sentinel or a member of a cluster headed by a neighbor sentinel.

The second step is *path discovery*. A small number of packets are sent from each sentinel towards the sink via the underlying routing protocol. For a sentinel, the *downstream clusters*¹ are defined to be the clusters towards which a packet is sent and *upstream*, the clusters from which packets are received. By overhearing the (one-hop) communications of its cluster members, the sentinel builds a table of the downstream and upstream clusters; thus, it may be able to periodically send traffic intensity updates towards the *upstream* path. This table is updated when regular traffic packets are sent during the normal network operation.

Updates by the cluster members are sent to the sentinels periodically (*i.e.*, proactively). Because during periods of quiescence the intensity level is low, sensors and sentinels send updates only after a threshold for the local load ρ and γ , respectively, has been exceeded. Thus, energy is saved and the proactivity level can be tuned. Message delivery is assumed to be handled by the routing protocol. The assumption is that the routing protocol allows sensors to discover and store information with regard to their neighborhood. Update delivery optimizations are left to the design of the routing protocol although the utilization of the Zone Routing Protocol (ZRP) [6], that combines proactive and reactive zone routing, is highly suitable and practical.

¹Clusters are identified by their sentinel's ID.

3.2 Distributed Traffic Intensity Estimation

In this subsection, we provide a methodology for approximating the traffic intensity. The traffic intensity is collectively estimated across multiple clusters. The estimation is based on a queuing network model, wherein each sensor is modeled as a queue. The queues are then interconnected within a cluster to form a network. It is assumed that the resulting network is a *BCMP* network of queues, which enables us to represent the state of the network in a product form [7]. This approximation model provides us with *macroscopic* network statistics allowing us to deal with the complexity of effectively estimating congestion.

In this approximation, the traffic intensity γ is defined as the steady state probability of at least one packet existing in the queueing network and is calculated as[7]:

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

where $P(n_1, n_2, \dots, n_i, \dots, n_N)$ is the steady state probability that the population in queue i (of N queues) is n_i and ρ_i is the load for queue i being defined as incoming packet rate over the service rate, *i.e.*, $\rho_i = \frac{\lambda_i}{\mu_i}$, and can be computed locally at each sensor. An analysis on the value of γ shows us that for increasing load ρ_i when $\gamma_{thres} = 0.95$ the cluster is considered to be congested.

3.3 Modeling the Traffic Intensity

Modeling how the traffic intensity values change with varying rates allows for the allocation and adjustment of rates to the sources within a cluster. Finding an exact expression for γ is very hard since we can make no assumptions regarding the distributions of the incoming and the service rates. We develop another approximation model which

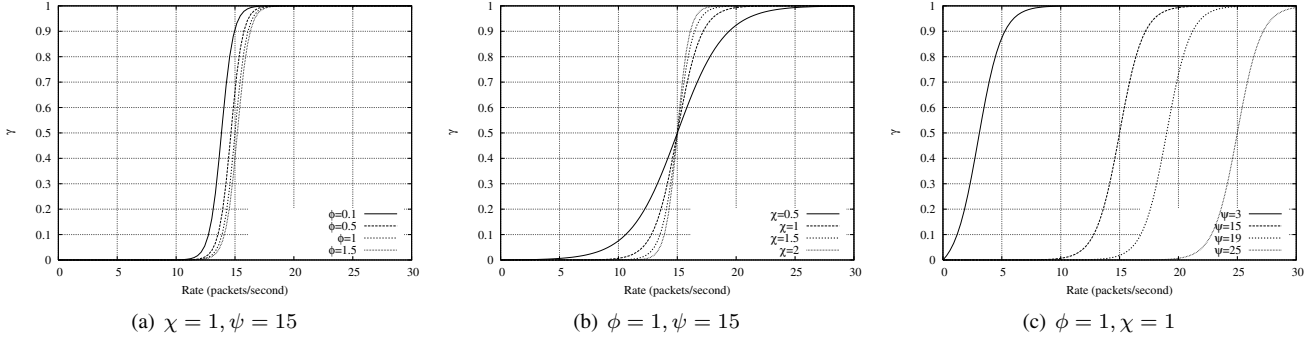


Figure 3. Effect of Parameters on the Model.

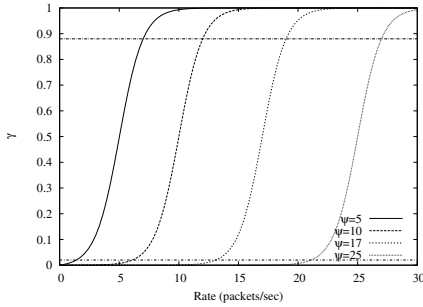


Figure 4. Examples of Tuning ψ .

seems to provide an accurate representation of the behavior of the traffic intensity.

To extract the approximate traffic intensity model, we look at its properties. We present these properties, using a sample experimental result shown in Figure 2, where two flows are initiated at the same time from clusters 1 and 2. We trace the flows' paths and find that the first flow traverses Clusters 2, 4 and 5 to reach the sink and the second flow traverses Clusters 1, 3, 4 and 5. We measure the γ value calculated within each cluster with increasing sending rates. We present our observations below:

- The resulting experimental outcome shows that the estimator curve follows a *sigmoid* function². We can thus try to model the estimator's values with changing rate with such a function.
- The estimation interval t_e was 3 seconds. We see that at the end of the estimation interval, different clusters have different estimated values (*i.e.*, the curves are different). This tells us that the perceived traffic intensity at each cluster may vary, *i.e.*, the curve modeling the traffic intensity at a specific point in time may differ. As traffic load increases, the queuing delay will

²The value of γ increases slowly at low rates but fast as the rate approaches the congestion value.

increase at the interfering clusters and cause *later in time*, packets from queues prior to the congested cluster to also be delayed and eventually result in the building up of these queues. As we show below, model parameters can be tuned to consider this effect.

Based on our first observation, we try to model the γ curve as a sigmoid function which can be represented as follows:

$$f(r) = \frac{1}{1 + \phi e^{-\chi(r-\psi)}} \quad (1)$$

where r represents the total rate observed for the collection of flows considered in the model. Parameters $\phi > 0, \chi > 0, \psi > 0$ define the details of the shape of the curve and can be fine tuned to better approximate the estimator's behavior.

Now, given Equation 1, we know that the model must return the value '0' for a rate equal to '0'. Also, the Traffic Intensity has an asymptote at $\gamma = 1$. The final model can be, then, represented as:

$$\gamma(r) = \frac{f(r) - f(0)}{1 - f(0)}, r \geq 0 \quad (2)$$

We plot Equation 2 in Figure 3 for different parameters. Other parameters could also be considered. We find that the parameter that most affects the outcome of the curve, and thus would mostly affect our estimation of the required values, is ψ . We use *Parameter Tuning* to decide the value of ψ which defines the most appropriate sigmoid curve to be advised at different point in time. Remaining parameters are set as predefined constant values.

We apply Parameter Tuning based on our second observation. Parameter Tuning can be done periodically and locally at the sentinel using the received updates concerning current flows on the path(s) that the cluster is involved in. Its goal is to utilize Equation 2 to calculate R_{est} as defined in Section 2 and appropriately allocate it to the sources within each cluster using the Rate Allocation Policy.

More specifically, assume that we have received a specific value for the total rate R_{tot} allocated to flows along a path. Assume that we are also provided with the estimation of γ . We can then use Equations 3 below which are transformations of Equation 2, solved for ψ and r respectively, first to calculate ψ for the specific point in time and then to calculate R_{est} for a desirable value of γ . In the solution for ψ , set $r = R_{tot}$ while in the solution for r , the result represents R_{est} .

$$\psi = \frac{\ln \frac{1-\gamma-e^{-\chi r}}{\gamma \varphi e^{-\chi r}}}{\chi}, r = -\frac{\ln \frac{1-\gamma}{\gamma \varphi e^{\chi \psi} + 1}}{\chi} \quad (3)$$

We provide two examples of how Parameter Tuning can be done (shown in Figure 4). In the first example we receive a total rate of 6 packets/sec and γ is estimated at 0.02. Thus, using the values in Equations 3, we get the value of ψ . We can now “predict” what would be the sustainable rate for a requested γ_{thres} of, say, 0.95 (i.e. the congestion threshold). Therefore, we may decide on whether it would be advisable to allow a new flow.

In the second example, the value of γ is near 0.9 at a rate of 19 packets/sec. We may again calculate ψ and choose a reduced rate in order to achieve a value of, say, 0.6, for γ (as it pertains to an uncongested scenario).

3.4 Rate Allocation Policies

We implement two policies for rate allocation based on admission control and proportional rate allocation. These policies can be customizable and can be plugged-in to the CoBRA framework to define how R_{est} should be allocated to flows of different classes. The Rate Allocation Policy is implemented at the sentinel and executed periodically to adjust the rates of the cluster sensors that are sources of traffic. A source may implement its own policy with respect to the received γ value. For example, a rate control scheme may be implemented to reduce the source rate when a specified γ_{thres} has been exceeded.

With our framework, two basic Rate Allocation Policies are implemented at the sentinels: (i) proportional allocation of the estimated maximum aggregate rate, and (ii) admission control based rate allocation.

Proportional Rate Allocation: In the Proportional Rate Allocation policy, the proportion of the rate assigned to each flow is decided by a weight $w(c_i)$ per class c_i , i.e., each class is represented by a weight. Note that proportional sharing is done among the classes that interfere due to coupled paths. Each set of interfering flows is treated independently. Proportional rate allocation implies that flows of the same class get the same proportion of R_{est} .

The algorithm is described below. Let a_i represent the rate assigned to flow i and R_{tot} be the aggregate rate for all

active interfering flows as defined in Section 2. (R_{tot} is initialized to ‘0’). The mechanism described below is a simple example of an application of our framework and could be extended to implement more sophisticated algorithms. The algorithm works as follows:

- $\gamma < 0.95$: Given an incoming flow i , assign the rate requested r_i to that flow independently of its class; $a_i = r_i$. Next, update R_{est} . To do this, retrieve the current estimated value of ψ , using γ and R_{tot} , as in Equation 3.
- $\gamma > 0.95$: This signifies congestion. The rates cannot be allocated in full and adjustment needs to be performed. For each of the active flows (say i) including the new flow, and n_j active flows of class j , allocate to flow i rate a_i as: $a_i = \frac{w(c_i)}{\sum_j n_j \cdot w(c_j)} \cdot R_{est}$.

Admission Control: In the admission control policy, each sensor within a cluster, before it initiates a flow, will have to predict whether the introduction of a new flow will congest the network. This is done by using the framework’s computed traffic intensity, γ , given the total rate that will result if the sensor initiates the flow.

As mentioned earlier, a flow may request to enter the network multiple consecutive times. Thus, it may eventually be able to enter when another flow completes and frees up network resources. This might cause a source to experience extended waiting times before being admitted. However the admission algorithm might be adjusted to accommodate additional weighting factors which can in addition account for the waiting times. The algorithm is described below:

For a newly created flow:

- Update R_{est} . To do this we retrieve current estimated the value of ψ using the observed γ and R_{tot} .
- Given the requested rate r_i , calculate the new value of γ with rate equal to $R_{tot} + r_i$.
- If the calculated $\gamma < 0.95$ (the aggregate rate is less than the value of R_{est}) admit the flow and assign $a_i = r_i$; else reject the flow.
- Repeat the process periodically.

4 Evaluation

We have implemented and tested our framework within Network Simulator *ns-2* [2]. We generated for each experiment, 5 random network topologies in a square area of 100m x 100m and have calculated average values. We varied the network density using different numbers of sensors (60, 120 and 200 sensors). One of the sensors was selected at random to be the sink. The simple CSMA-based

MAC protocol, with an exponential backoff policy was assumed; this has been generally used in prior work on sensor networks [16][19][18] that do not specifically address the MAC layer.

The sensor nodes are set up to simulate the characteristics of MICA motes [1]. They are homogeneous and have a transmission range of $25m$. Data packets are 30 bytes (the standard size). The memory in a mote is 4KB; thus, the queue size is set to a size 65 packets. We assume the presence of robust physical layer techniques to cope with bit errors; hence, packet drops are attributed only to queuing related drops and to collisions. We have simulated a scenario where multiple consecutively occurring events are sensed at random points on the sensor field.

4.1 Overall Load Adjustment

We compared CoBRA with a very popular congestion/rate control technique used in [10, 19], *i.e.*, AIMD (Additive Increase – Multiplicative Decrease) with reactive backpressure, in which messages are forwarded all the way to the source upon the detection of congestion. In this, congestion is identified when a queue of a sensor overflows. We also compare CoBRA to a *basic* approach in which sources start sending with a specified rate without any rate and congestion control algorithms employed. In our comparisons, we vary the *maxrate* value for AIMD, *i.e.*, the maximum rate a source can send during the additive increase. Rates are dropped to half while congestion persists.

We have evaluated CoBRA using the proportional rate allocation policy, described in Section 3.4, where we set all flows to be of the same class. We vary the requested rate for the sources in our experiments. We set $\gamma_{thres} = 0.95$. We have experimented with random networks of sizes 60, 120 and 200 nodes in order to change the network density. Due to space limitation we present our results for 200 nodes, a test case for dense networks, and 60 nodes, a test case for sparse networks.

Our first experiment was to measure the per-hop delay metric. Per-hop delay is a fairer metric than the end-to-end delay because flows may be initiated close or further away from the sink. We first show, in Figures 5 and 8, that both AIMD and CoBRA achieve lower per-hop delays than the basic approach (no policy used for rate control). CoBRA performs better than AIMD at higher rates, *i.e.*, at higher loads. This is because AIMD is reactive and packets that have reached the queues when overflow was observed, experience high delays during the time backpressure messages are forwarded towards the sources. We also notice that the improvement in per-hop delay is more significant with higher density networks.

With respect to the total throughput, we observe in Figures 6 and 9, that CoBRA manages to deliver about 10%

Flow ID	Rate(p/s)	Start Time	Weight
1	35	0	1
2	35	5	2
3	35	10	3
4	35	15	1
5	35	20	2

Table 1. Flow Setup (Proportional Rate Allocation).

more packets per second than AIMD and the basic approach. This is because CoBRA avoids the additive increase phase while the reduction in rate when congestion occurs is smoother. In addition, AIMD will cause packet drops and thus, a reduction in throughput is caused when the sources reach high rates during additive increase. CoBRA, on the other hand, provides an adaptive aggregate rate estimation which allows smooth adjustments of the reallocated rates to sources.

Finally, in Figures 7 and 10, we see that besides delivering more packets than AIMD, CoBRA also achieves a slightly higher delivery ratio. We note that the higher delivery ratio reflects also *energy savings*, since much less energy is wasted due to packets dropped en route the sink.

4.2 Evaluation with Proportional Rate Allocation

In our second set of experiments, our attempt is to evaluate how well R_{est} is estimated and how rates are allocated to the flows. We concentrate on dense networks of 200 nodes. We implement the proportional rate allocation scheme described in Section 3.4, in which the specific class of a flow is reflected by its *weight*. The weight of a flow defines the relative proportion of R_{est} that the flow should be allocated. The algorithm periodically recalculates R_{est} and re-assigns rates at each cluster.

We start five flows consecutively, one every 5 seconds at random positions in the network. In our experiments, our observation was that all flows eventually interfere with each other either near or further from the sink. We have deliberately selected a requested rate of 35 packets/sec. In Figure 5 we saw that with the basic technique (no congestion-control algorithm), when all flows send with a rate of about 20 packets/sec, the packets start to experience very high and arbitrary delays which signifies congestion. We have chosen the rate of 35 packets/sec to ensure that no two flows will receive full rate without causing congestion. Finally, weights are assigned as shown in Table 1.

In Figure 11 we plot the allocated rates throughout the simulation execution per flow. Initially, Flow 1 receives the full requested rate as the calculated γ remains below γ_{thres} .

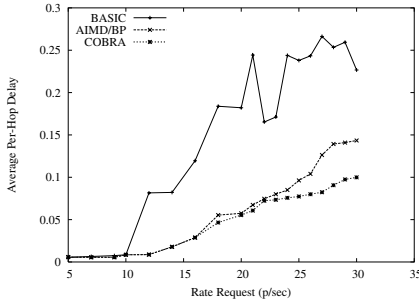


Figure 5. Average Per-Hop Delay (Dense Network).

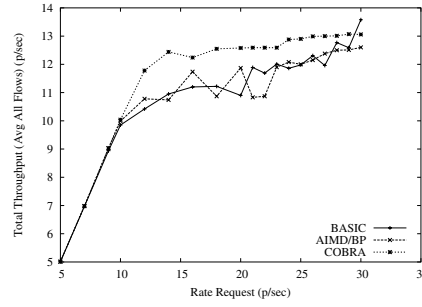


Figure 6. Throughput (Dense Network).

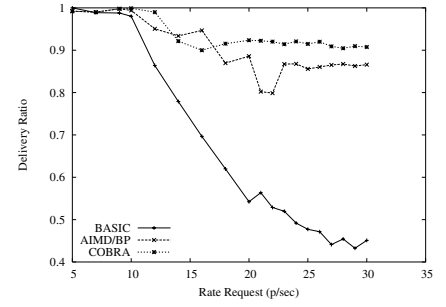


Figure 7. Delivery Ratio (Dense Network).

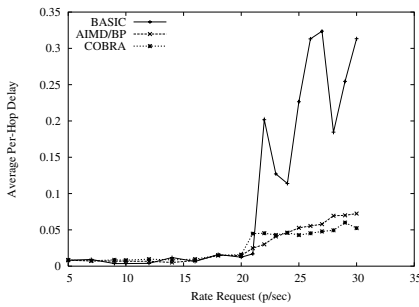


Figure 8. Average Per-Hop Delay (Sparse Network).

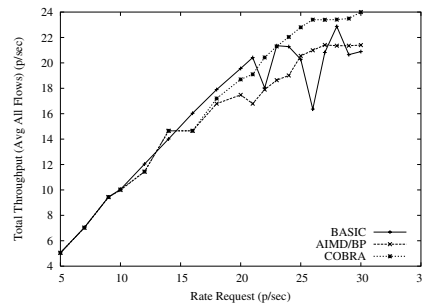


Figure 9. Throughput (Sparse Network).

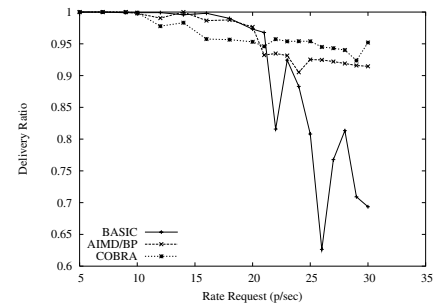


Figure 10. Delivery Ratio (Sparse Network).

When Flow 2 with weight 2 (*i.e.*, double than that of Flow 1) is introduced, the proportional rate assignment algorithm requires that Flow 1 should be assigned half the rate assigned to Flow 2. Therefore, a rate of about 17 packets/sec is re-allocated to Flow 1. Similarly, the introduction of flows 3, 4 and 5 forces the reassignment of rates to the sources.

Note that the fluctuations observed are caused because of the variations in the estimation of R_{est} at each source cluster. For example, when a new flow is introduced, the traffic increases suddenly and the traffic intensity is overestimated. Thus, the rate allocated is lower because of the underestimation of R_{est} . However, very quickly, ψ is re-calculated and the allocation of the rates is improved. We also note that the estimator provides more accurate feedback (less fluctuations) as the number of flows increases.

Figure 12 shows that CoBRA compared to AIMD with backpressure, performs better in terms of the resulting average per-hop delay. CoBRA is more conservative than AIMD in terms of the total allocated rate. On the other hand, it avoids additive increase which might cause congestion at the point where the rate reaches the maximum value, even if sources do not reach the maximum simultaneously.

In Figure 13, we notice that CoBRA manages to maintain fair total throughput for the flows with respect to their weights whereas the AIMD/BP technique cannot ad-

equately adjust the throughput. This is because the multiplicative decrease when receiving a congestion backpressure message does not consider each flow's class.

Finally, in terms of delivery ratio, we observe in Figure 14 that both techniques achieve very high ratios (over 85% in almost all cases). We do notice however, that the low weight flows 1 and 4 have lower delivery ratios. This is because, fewer packets are received from these sources (with CoBRA), due to the proportional rate sharing.

4.3 Evaluation with Admission Control

In the third set of experiments we test the ability of our framework, when calculating R_{est} , to allow or disallow the initiation of a new flow, while doing this on a localized, cluster level.

We start 5 flows, one every 5 seconds with the rate requests shown in Table 2. One of the flows is stopped at some specific point in time and frees the allocated resources. In Figure 15 we plot the assigned rates for the admitted flows. Flows 1, 2 and 3 are admitted. When Flow 4 attempts to enter, the sentinel rejects the flow as the intensity estimation exceeds the preset threshold of 0.95. This is also the case for Flow 5. However, as soon as Flow 1 stops, Flow 4 reattempts to enter and is admitted shortly after. Notice that the

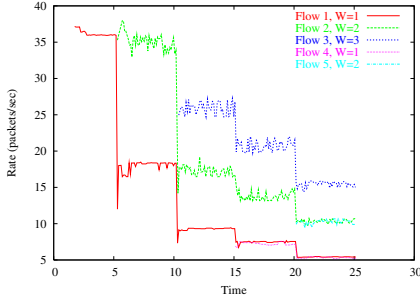


Figure 11. Proportional Rate Allocation.

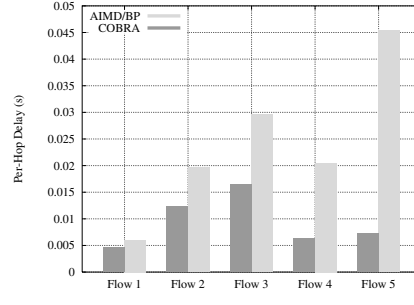


Figure 12. Average Per-Hop Delay Per Flow.

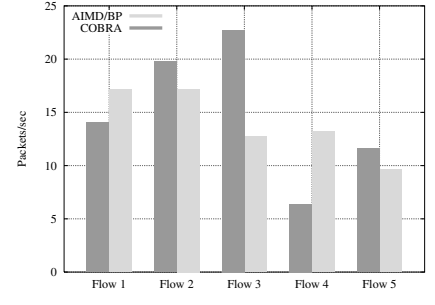


Figure 13. Total Average Throughput per Flow.

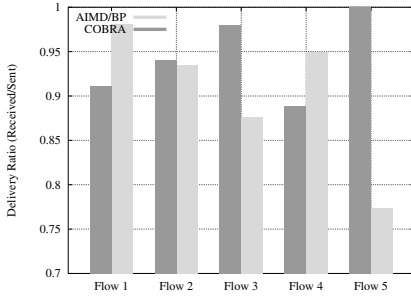


Figure 14. Delivery Ratio Per Flow.

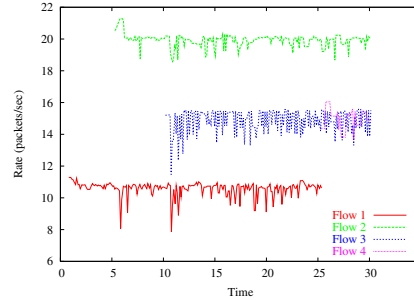


Figure 15. Rate Allocation For Admitted Flows.

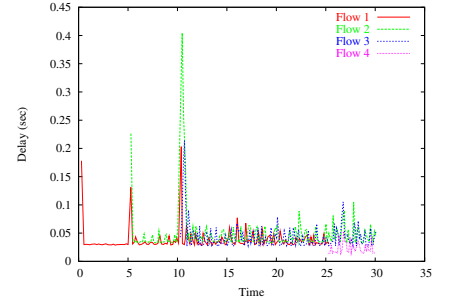


Figure 16. Per-Hop Delay for Admitted Flows.

Flow ID	Rate(p/s)	Start Time	End Time
1	10	0	25
2	20	5	30
3	15	10	30
4	15	15	30
5	20	20	30

Table 2. Flow Setup (Admission Control).

rates fluctuate slightly (around ± 0.5 packets/sec and very rarely as much as -2 packets/sec). This is observed especially when a flow is initially admitted and is attributed to the fact that the flows are not completely smooth and therefore the intensity estimates may fluctuate. The rate policy could be further modified to start a flow only when the exact rate can be archived and stop it otherwise at each (periodic) rate allocation instance. Finally, in Figure 16 we plot the per-hop delay. We notice that the delay is kept mostly at very low levels ($< 0.05s$), while this is true for all flows. This means that our framework manages to control the load without irregular delays among different flows. A small number of packets may experience higher delay occasionally; more precisely when a new flow is admitted and the Traffic Intensity Estimation has not yet been fine-tuned.

4.4 Energy Cost

In our last experiment we measure the energy savings of using CoBRA. Our goal is to study whether the usage of our techniques is beneficial with respect to the overhead messages produced during the proactive updating phase and the benefits gained due to reduction in packet drops. As discussed in Section 3.1, updates are triggered when ρ and γ exceed a specific threshold. The energy wastage, W , is measured as the total number of bytes transmitted for updates, U , plus the total number of bytes transmitted in packets that are eventually dropped, D , i.e., $W = U \cdot hops_U + D \cdot hops_D$. The parameters $hops_U$ and $hops_D$ represent the total number of hops that the packets were forwarded. Data packets for CoBRA and AIMD are $s_d = 30$ bytes, update packets for CoBRA are also $s_u = 30$ bytes while backpressure messages with AIMD are $s_{bp} = 13$ bytes.

For AIMD/BP, U is equal to the number of backpressure messages sent, times the size of the backpressure message i.e., $U_{AIMD} = N_{BP} \cdot s_{bp}$. For CoBRA, U is equal to the number of sentinel and sensor updates, times the size of the update message i.e. $U_{CoBRA} = (N_{SENTINEL} + N_{SENSOR}) \cdot s_u$. For both cases, D is equal to the number of dropped packets, times the data packet size $D = N_{DROPPED} \cdot s_d$. Note that sensor updates are always sent

at one hop. We can, then, calculate W_{AIMD} and W_{CoBRA} which we plot for a network size of 200 nodes.

In Figure 17 we show the total number of overhead bytes broken down in terms of control and dropped packets. Control packets for CoBRA include sensor and sentinel updates while for AIMD, only include backpressure messages. We notice that as the rate request increases (and consequently the load that is attempted to be injected into the network) CoBRA produces more update messages and thus has higher overhead with respect to this type of messages. Also note that when control packets are low for CoBRA, the wasted bytes due to dropped packets are higher because the proactive phase does not take full effect. As the rate request increases, AIMD cannot effectively handle congestion in a timely manner and the wastage due to dropped packets dominates over the control packet overhead. We reiterate here that the proactive process is started when specific thresholds for the traffic load are exceeded, a fact with additional positive effects: Due to its adaptive nature, CoBRA regulates congestion, thus, controls the load. This results in limiting the need for control packets during the proactive update process. The overall overhead is shown in Figure 18, where we observe that the benefits gained with CoBRA due to the reduction of dropped packets compensate for the proactive updates.

5 Related Work

Resource management for real-time traffic has been mostly studied in the area of wireless ad-hoc networks. In [5] a resource reservation strategy is proposed based on the EDF scheduling policy. This work concentrates on admission control at the MAC layer. In [17], QoS soft guarantees are achieved, but as in our work, no MAC layer scheduling is assumed. However, only single-hop networks are considered. In [13] the authors attempt to solve the problem of fair bandwidth assignment for *multihop* wireless networks. Unlike our work, the goal is to maximize end-to-end throughput via maximization of the spatial reuse of the spectrum.

Further research on resource reservation include the SWAN protocol [3]. SWAN is a simple, stateless end-to-end resource reservation mechanism to support real-time and best effort traffic. Although in sensor networks simplicity is desirable, end-to-end solutions might not be responsive enough, especially in highly unpredictable event-based environments. In [21], a localized admission control technique is proposed which considers the effects of multiple interfering flows. However, these techniques might be unsuitable for sensor networks due to their complexity, especially as multiple classes need to be supported.

Scheduling messages with deadlines has been the problem dealt with in [9]. This work provides an efficient, centralized algorithm for communication scheduling that

avoids delays due to contention. MAC layer prioritization techniques have also been proposed in [20] while bursty traffic support (also at the MAC layer) was studied in [22].

Congestion Control techniques are used to control the network load level. In [11], a technique is proposed that differentiates between low and high importance packet flows. Rate allocation and control is, however, only limited to AIMD-like techniques. In our work, instead, we propose a *generic* framework utilizing traffic intensity modeling to allow implementation of different rate allocation algorithms. CODA [19] is another congestion control mechanism for sensor networks that employs end-to-end and backpressure techniques to reduce congestion. The technique is reactive and may not avoid dropping a large number of packets when multiple flows are introduced simultaneously. Other works such as [10] and [4] study the effects of congestion on realistic tree structured, sensor deployments and propose solutions for relieving congested areas.

SPEED [8, 15] is a routing protocol developed to support real-time packet delivery in sensor networks. It is based on the idea of maintaining a specific traffic velocity along the path given an initial deadline. Although the simplicity of SPEED makes it an attractive solution, it does require topology information; congestion is still dealt with in a reactively manner by rerouting packets away from the hotspot areas.

Rate control techniques have also been utilized to avoid overload. ESRT [16] is a protocol for rate control in sensor networks aiming to achieve a specific level of reliability. The reliability metric is used as a parameter for adjusting the rates in the presence of congestion. ESRT requires a centralized sink to adjust the rate for *all* the sensors in the network and is better suited for monitoring applications.

6 Conclusions

In this paper we have presented CoBRA, a framework that handles congestion and enforces rate control, to support multiple classes of flows in sensor networks. CoBRA provides collective and distributed network traffic intensity estimation and modeling of its behavior. This model can be used within clusters of sensors to make localized decisions as to how rates can be assigned to active or prospective flows in order to maintain a desirable network load level. CoBRA supports rate allocation algorithms. We implement a simple proportional rate allocation algorithm using our framework and show, via simulations, that it is fairer and more responsive in terms of delay than traditional, reactive AIMD/BP techniques. We also implement admission control showing that our estimations can handle randomly initiated flows of multiple classes. For our future work we intend to study the performance of CoBRA over different routing protocols.

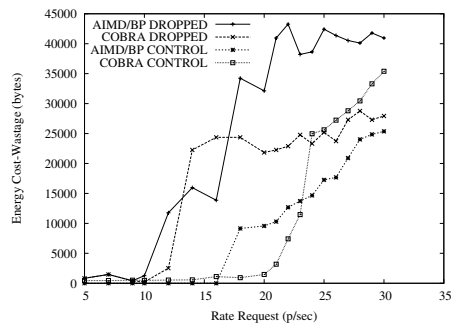


Figure 17. Overhead Brake-down.

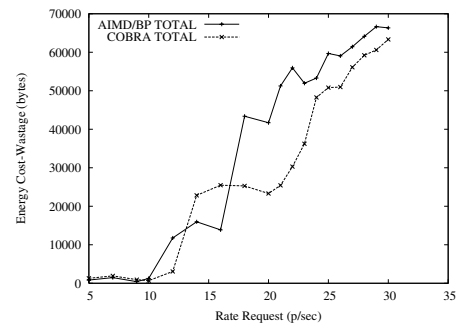


Figure 18. Total Overhead.

References

- [1] Crossbow MICA motes. Available at www.xbow.com.
- [2] NS-2. Available at www.isi.edu/nsnam/ns.
- [3] G.-S. Ahn, A. T. Campbell, A. Veres, and L.-H. Sun. Swan: Service differentiation in stateless wireless ad hoc networks. In *Proc. of the 21st Joint Conference of the IEEE Computer and Communications Societies*, pages 457–466, New York, NY, Jun. 2002.
- [4] C. T. Ee and R. Bajcsy. Congestion control and fairness for many-to-one routing in sensor networks. In *Proc. of the 2nd international conference on Embedded networked sensor systems*, pages 148–161, Baltimore, MD, Nov. 2004.
- [5] T. Facchinetti, L. Almeida, G. C. Buttazzo, and C. Marchini. Real-time resource reservation protocol for wireless mobile ad hoc networks. In *Proc. of the 25th IEEE Int'l Real-Time Systems Symposium*, pages 382–391, Lisbon, Portugal, Dec. 2004.
- [6] Z. J. Haas and M. R. Pearlman. The zone routing protocol for ad hoc networks. Internet Draft, 1998.
- [7] J. F. Hayes and T. Babu. *Modeling and Analysis Of Telecommunication Networks*. 2nd edition. Wiley-Interscience, New Jersey, NJ, 2004.
- [8] T. He, J. A. Stankovic, C. Lu, and T. F. Abdelzaher. SPEED: A stateless protocol for real-time communication in sensor networks. In *Proc. of the 23rd Int'l Conf. on Distributed Computing Systems*, pages 45–55, Tokyo, Japan, May 2003.
- [9] P. S. Huan Li and K. Ramamritham. Scheduling messages with deadlines in multi-hop real-time sensor networks. In *Proc. of 11th IEEE Real Time and Embedded Technology and Applications Symposium*, pages 415–425, San Francisco, CA, Mar. 2005.
- [10] B. Hull, K. Jamieson, and H. Balakrishnan. Mitigating congestion in wireless sensor networks. In *Proc. of the 2nd international conference on Embedded networked sensor systems*, pages 134–147, Baltimore, MD, Nov. 2004.
- [11] K. Karenos, V. Kalogeraki, and S. V. Krishnamurthy. 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*, pages 107–114, Sydney, Australia, May 2005.
- [12] S. Lee, G. Ahn, X. Zhang, and A. T. Campbell. INSIGNIA: an IP-based quality of service framework for mobile ad hoc networks. *Parallel Distributed Computing*, 60:374–406, 2000.
- [13] B. Li. End-to-end fair bandwidth allocation in multi-hop wireless ad hoc networks. In *Proc. of 25th IEEE Int'l Conf. on Distributed Computing Systems*, pages 471–480, Columbus, Ohio, Jun. 2005.
- [14] J. Li, C. Blake, D. S. D. Couto, H. I. Lee, and R. Morris. Capacity of ad hoc wireless networks. In *Proc. of the 7th int'l conf. on Mobile computing and networking*, pages 61–69, Rome, Italy, Jul. 2001.
- [15] C. Lu, B. M. Blum, T. F. Abdelzaher, J. A. Stankovic, and T. He. RAP: A real-time communication architecture for large-scale wireless sensor networks. In *Proc. of the IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 55–66, San Jose, CA, Sep. 2002.
- [16] Y. Sankarasubramaniam, B. Akan, and I. F. Akyildiz. Esrt: event-to-sink reliable transport in wireless sensor networks. In *Proc. of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 177–188, Annapolis, MD, Jun. 2003.
- [17] S. H. Shah, K. Chen, and K. Nahrstedt. Dynamic bandwidth management in single-hop ad hoc wireless networks. *Mobile Networks and Applications*, 10:199–217, 2005.
- [18] C. Y. Wan, A. T. Campbell, and L. Krishnamurthy. PSFQ: A reliable transport protocol for wireless sensor networks. *IEEE Journal on Selected Areas in Communications*, 23:862–872, 2005.
- [19] C.-Y. Wan, S. B. Eisenman, and A. T. Campbell. Coda: congestion detection and avoidance in sensor networks. In *Proc. of the 1st international conference on Embedded networked sensor systems*, pages 266–279, Los Angeles, CA, Oct. 2003.
- [20] X. Yang and N. H. Vaidya. Priority scheduling in wireless ad hoc networks. In *Proc. of the 6th ACM Int'l Symposium on Mobile Ad Hoc Networking and Computing*, pages 71–79, Urbana-Champaign, IL, May 2002.
- [21] Y. Yang and R. Kravets. 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*, pages 379–388, Fort Lauderdale, FL, Oct. 2004.
- [22] H. Zhu and G. Cao. On improving service differentiation under bursty data traffic in wireless networks. In *Proc. of the 23rd Conf. of the IEEE Communications Society*, volume 2, pages 871–881, Hong Kong, Mar. 2004.