Congestion Control for Spatio-temporal Data in Cyber-physical Systems

Hossein Ahmadi, Tarek Abdelzaher, Indranil Gupta
Department of Computer Science
University of Illinois at Urbana-Champaign
hahmadi2@uiuc.edu, {zaher, indy}@cs.uiuc.edu

***Abstract*—Data dissemination protocols in cyber-physical systems must consider the importance of data packets in protocol decisions. Importance of data cannot generally be accurately represented by a static priority value or deadline, but rather must stem from the dynamic state of the physical world. This paper presents a novel congestion control scheme for data collection applications that makes two key contributions. First, packet importance is measured by data contributions to the accuracy of estimating the monitored physical phenomenon. This leads to congestion control that minimizes estimation error. Second, our protocol employs a novel mechanism, i.e. spatial aggregation, in addition to temporal aggregation to control congestion. The protocol is generalized to multiple concurrent applications. Our approach employs different granularities of aggregation in transporting spatio-temporal data from nodes to a base station. The aggregation granularity is chosen locally based on the contribution of the transmitted data to the reconstruction of the phenomenon at the receiver. In an area affected by congestion, data are summarized more aggressively to reduce data transfer rate while introducing minimal error to the estimation of physical phenomena. We implement this scheme as a transport layer protocol in LiteOS running on MicaZ motes. Through experiments, we show that the proposed scheme eliminates congestion with an estimation error an order of magnitude smaller than traditional rate control approaches.**

I. Introduction

Traditional embedded and control systems typically *close loops* around important physical phenomena. More recently, wireless sensor networks emerged where the sensing function is largely distributed and the emphasis is more on phenomenon *estimation*, rather than *control*. From a cyber-physical perspective (where a key goal is to understand how couplings between the cyber and physical realms affect system design), sensor networks motivate analysis of how network protocols that communicate sensor data affect phenomenon estimation accuracy, and in turn how this knowledge can improve network protocol design. Accordingly, this paper redesigns congestion control functions in sensor network transport protocols to account for importance of physical data. We show that by accounting for the impact of communicated sensor data on estimation accuracy in congestion control decisions, the error in estimating aggregate phenomena can be reduced by as much as one order of magnitude.

Research was sponsored in part by Natural Sciences and Engineering Research Council of Canada (NSERC), NSF Grant CNS 06-26342, and by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

A sample wireless sensor network can be a set of wireless nodes in a surveillance area that are equipped with appropriate sensors to monitor temperature and humidity distributions. The sampling can be performed with a fixed rate (e.g. once every second). Each sample is called *spatio-temporal* since it represents a value in space and time. Suppose that two different applications transport temperature and humidity data in a multi-hop fashion to base stations for analysis and modeling. Congestion occurs when a node is not capable of forwarding all the samples it receives to the next-hop node.

In our previous work [1], we proposed a transport protocol with *adaptive reliability*, where the number of retransmissions (of lost packets on an unreliable wireless medium) is adapted based on the expected error in estimating a physical phenomenon caused by the data loss. This paper significantly differs from that work in two important respects. First, we redesign *congestion control* functions, as opposed to *reliability* and retransmissions. A new congestion control scheme is designed and implemented for data collection networks that maximizes estimation accuracy in the face of congestion. While our previous work on reliability focused on probabilistically (re)transmitting data to overcome link failures, here we discuss the problem of controlling data flow through *intelligent aggregation*. Second, our protocol is extended to handle multiple concurrent applications of different importance. Our prior work on reliability treated all traffic alike.

Many congestion control protocols have been proposed for wireless sensor networks [2], [5], [8], [12], [17]–[19], [22], [24]. Modifying the data generation rate is the main mechanism that has been widely used in the transport layer. Routing around the congestion is usually employed as a network layer or cross-layer mechanism. Decreasing the data rate implies reducing the number of spatio-temporal samples received by the base station and therefore increasing the estimation error. Previous efforts, motivated by layering concerns, incorporate only limited knowledge of the real importance of a data packet into protocol decisions. So, they cannot control the accuracy of the estimation when decreasing data rate (and

thereby decreasing the number of spatio-temporal samples). This means that even limited congestion on one node may result in increasing estimation error considerably. Fairness is traditionally provided by allocating the same bandwidth to all concurrently running applications.

This work alleviates the above shortcomings by taking estimation error into account, and allowing different applications to have different estimation errors depending on their relative importance. The relative ratio of the estimation errors of different applications is user-defined. In the special case, the user can enforce that all applications experience the same estimation error.

Finally, as alluded above, previous protocols resolve congestion by decreasing source rate which forces the source to either fall behind or do *temporal* summarization of data (i.e., send aggregates of samples at a lower rate). This temporal summarization at the source is not efficient because it uses no information about readings from other nodes (i.e., spatial information). In contrast, we resolve congestion by using both spatial and temporal information. Hence, we effectively allow for better data compression. Our scheme is independent of the type of space used in summarization, i.e. whether it is abstract space (e.g. node ids) or physical space (e.g. node locations), although we use abstract space aggregation in this work. Summarization in space and time with variable granularity is used as a tool to control the data transfer rate and eliminate congestion. The degree of summarization depends on the estimation error it contributes to the overall estimation accuracy. Using least-error summarization, our scheme eliminates congestion while incurring the least possible overall error in sensing the physical environment.

We further show through analysis that our protocol has a small processing overhead, enabling deployment on very CPU-constrained motes. When several applications share the same network, we prove that our scheme provides each with the assigned share of the resources. This protocol is implemented as a transport protocol in LiteOS on a testbed of MicaZ motes. Through several experiments, we demonstrate that our protocol eliminates congestion promptly while providing weighted fairness. We implement two well-known congestion control protocols in LiteOS and compare them against ours. The results show that our protocol leads to a much smaller estimation error at the base station.

The rest of this paper is organized as follows: In Section II, we summarize the related work. Section III describes the congestion control mechanism and system model used. In Section IV, we formally validate our approach. Next, we evaluate our protocol and compare it against other congestion control protocols in Section V. Finally, we conclude the paper in Section VI.

II. Related Work

Congestion control is widely studied in general multi-hop wireless networks and many protocols that are designed for end-to-end data delivery [11], [21]. RCP [11] is designed for wireless networks with heterogeneous interfaces, where the receiver identifies lossy links and controls the transfer rate in the source to achieve high throughput. Another work [21] studies the main design issues in TCP variants such as the window-based transmission and distinguishing between congestion and link failure and proposes the ATP protocol. Such protocols cannot efficiently be applied to data collection in cyber-physical systems with many-to-one traffic.

Several efforts specifically address congestion control in wireless sensor networks [2], [5], [8], [10], [12], [17]–[19], [22], [24]. Such protocols can be divided two three classes: i) protocols designed for node-to-node data transfer (e.g. [10], [12]), ii) centralized rate control for many-to-one data collection (e.g. [2], [5], [17], [19], [22]), and iii) distributed rate control for many-to-one data collection (e.g. [8], [18], [24]). We briefly review each category of related work and explain why our approach is different at the end.

The first class of protocols are centralized rate control protocols designed for node-to-node data transfer. STCP [10] is a general transport protocol mainly designed for point-to-point communication in sensor networks that allows the base-station to control both reliability and data transmission rate. *Flush* [12] is a reliable transport protocol that monitors interference to control sending rate. It proposes a novel dynamic rate control mechanism that uses measured inference information to calculate the best delivery rate.

Centralized rate control schemes for many-to-one data collection are the second class of congestion control schemes. Event-to-Sink Reliable Transport (ESRT) [19] assumes that a fraction of sensor readings are enough to correctly identify an event in the surveillance area. Thus, the base-station tunes nodes reporting frequency such that the required information is obtained with minimum energy spending considering one-hop communication between nodes and the base-station. A similar measure of reliability is proposed in PORT [25], where a function of the received packet rate determines the reliability. Using feedback from the base-station, a certain threshold on packet rate is achieved while the total number of transmissions is minimized. In [22], authors study the congestion control problem not in the upstream direction, from sensor to sink, but in the downstream direction, from sink to sensor and they provide a rate control mechanism to reduce workload. The real-time and reliable transport protocol [7] focuses on several characteristics of different applications such as number of packets received from the event and the delay constraint of decision interval. Bian et al. [2] propose a protocol that assigns sending rates to all sensors in the routing tree in a centralized fashion. The rates are assigned based on the link characteristics present in the routing tree. Paek et al. [17] propose the rate controlled reliable transport protocol (RCRT) where the receiver is responsible for detection and explicitly determining sender rates. This provides flexibility to control the rate allocation over multiple senders.

The third category of protocols are distributed rate control schemes for many-to-one traffic. Congestion Detection and Avoidance (CODA) [24] employs two modes to eliminate congestion. In the back-pressure mode, a congested node

signals upstream nodes and pushes the congestion towards the source. An end-to-end congestion control using ACKs is another mode of congestion avoidance. A congestion control scheme for tree-based data collection has been proposed [5]. The rate control is done in a distributed manner where each node pushes the congestion towards the leaves by sending a maximum transfer rate to the upstream nodes. The rate is shared equally among upstream nodes and the process continues. Hull et al. [8] argue that congestion control in wireless sensor networks requires a range of mechanisms acting on different layers of the protocol stack. They propose *Fusion*, a combination of priority based media access control, hop-by-hop flow control, and source rate control. IFRC [18] is a distributed rate allocation scheme based on observed interference from the link layer. Through local decisions, IFRC maintains a fair allocation of medium to the contenders. To the best of our knowledge, there is *no congestion control scheme that considers true importance of data packets in terms of phenomenon estimation accuracy and provides an efficient yet simple protocol that can be implemented on motes*.

Data aggregation has been introduced for general sensor network applications [9], [15]. Directed Diffusion [9] is a data dissemination approach at network layer which allows multi-path forwarding and aggregation of the data. Paths are selected based on their latencies providing high efficiency when routing through multiple paths. Data aggregation on the other hand allows intermediate nodes to merge sensor data and reduce the data transfer rate. The Tiny Aggregation Service (TAG) [15] uses data aggregation in a querying service. The sensor network is treated as a database where SQL-like queries are performed to obtain data periodically. Previous work on aggregation often breaks traditional end-to-end congestion control semantics and congestion control schemes. None of the previous efforts have studied variable aggregation granularity for the purpose of efficient congestion control.

In real-time computing, several protocols have been proposed for sensor network communication [3], [6], [13], but they restrict themselves to different mechanisms for deadline or priority enforcement. Some application semantics, such as estimation error, do not lend themselves well to urgency-based prioritization. New schemes are needed to decide which packets to forward or how packet content is combined based on physical-world state. "Cyber-physical" data dissemination protocols (that consider physical-world state) have recently been described to enhance the performance of wireless sensor networks by utilizing physical-world knowledge about the sensor data. In our previous work [1], we proposed one such adaptive-reliability transport protocol. Based on the amount of error in estimating a physical phenomenon, nodes locally carry on a probabilistic retransmission scheme to achieve a guaranteed expected estimation error. In the current paper. we borrow the same protocol API and system model, but solve the new problem of controlling *congestion* while minimizing estimation error through intelligent and variable spatial and temporal aggregation. We hope that this protocol will serve as example for a new brand of protocols that address physical world state and cyber-physical coupling in a wide array of embedded network functions, beyond priority enforcement.

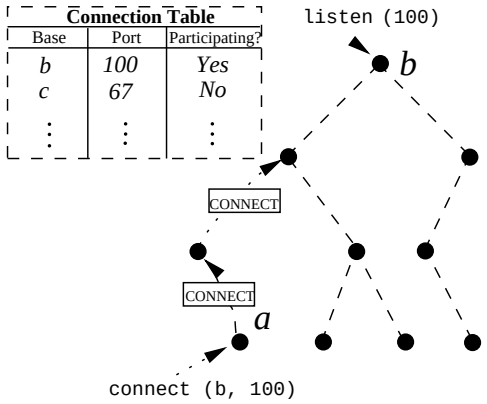


Fig. 1. Connection setup phase.

III. Cyber-physical Congestion Control

In data collection networks, the main challenge in designing a cyber-physical congestion control scheme (i.e., one where the coupling between the cyber and physical domains is accounted for) is to eliminate congestion without significantly increasing the global estimation error. This should be done with very little communication overhead. Moreover, different applications may have different accuracy requirements to be maintained. We aim to provide weighted differentiation among the applications based on user-defined accuracy weights. For example, if accuracy of application A is twice as important as accuracy of application B, then the estimation error weights of the two applications can be set 2:1. It defines the statistically expected ratio of these errors to be maintained by our congestion control scheme. Our congestion control achieves this ratio while minimizing the absolute error by summarizing data adaptively both in space and time.

Our approach is based on aggregation at each intermediate node. Each node sends an aggregate to its next hop periodically. The output rate of a node is controlled by summarizing the data to create smaller aggregates or averaging aggregate values of different report periods. The main idea of our protocol is to start with maintaining zero estimation error (due to summarization) at every node by forwarding every sensor reading to the base station. When congestion is detected, nodes in the congested area are allowed to cause some estimation error by reducing their data transmission to eventually eliminate congestion. We are interested in finding the minimal allowed error which is just sufficient to alleviate the congestion.

In this section, we first present the system model used in this paper. Next, we give an overview of the protocol explaining connection management and operation in non-congested situations. Finally, we discuss the mechanism for congestion control and describe the details of our adaptive summarization method.

A. System Model

Our congestion control components run locally on every node as a part of the transport layer. Several monitoring applications may run concurrently where for each application, a node called *base station* is responsible for collecting the data. The participating sensors perform the sensing and send the measurements to the corresponding base station for processing. Readings are aggregated at intermediate nodes and are forwarded to the parent node. To support this scenario, each application creates a transport layer connection. Each connection uses a convergecast tree obtained from the underlying routing protocol to collect data from the participating nodes. The tree is rooted at the base station and spans over a connected subset of nodes including the ones participating in monitoring as well as relay nodes. The connection is initiated at the base station. The network layer invokes the transport layer at intermediate hops as well as the destination.

Given this system model, congestion occurs when a node cannot transmit its (aggregate) data values at the rate they are generated. This happens if the bottleneck resource, such as communication bandwidth from a node to its parent, cannot accommodate the sending rate. Our protocol is agnostic to the nature of the bottleneck resource. For example, in energy-starved systems, congestion may be defined by the condition when the average energy consumption constraint at a node cannot accommodate the required sending rate. Using some energy control mechanism (e.g. [4]), transmission is delayed when budget is exceeded.

We represent a sensor measurement at a specific time and location by a sampling point in space and time. In a monitoring application, we expect the reconstructed physical phenomena to have the same values as those sensed by nodes at the corresponding time and location.

We use the following notation in the rest of this paper. For any given connection, we denote the routing tree by T and the base station by r. Let T_v be the sensor network subtree rooted at node v. Also, let $w_1, \ldots, w_k$ be successors of node v in T. Let $M_v(t) = \{m_x(t) | x \in T_v\}$ be the measured phenomenon in the region covered by T_v at time t where $m_x(t)$ is the actual reading of the sensor at node x. In our protocol, every node v has an estimation of space-time data collected in $T_z \subseteq T_v$. We define $\bar{M}_z^v(t) = \{\bar{m}_x^v(t) | x \in T_z\}$, to be the estimated values of phenomenon in the region T_z and at time t which is present to node v. $\bar{m}_x^v(t)$ is the value that node v locally thinks that x has measured. We formally define the estimation error as follows:

Definition 1 (Estimation Error): Estimation error at node v is defined as $E_v = \sum_{t_j} \sum_{x \in T_v} (m_x(t_j) - \bar{m}_x^v(t_j))^2$ where t_js are sampling times.

B. Protocol API and Non-congested Operation

The transport-layer API adopted by our congestion control protocol has previously been published in our own previous work [1]. To make the current paper self-contained, we review it below. Note that, we did not change the API for backward compatibility reasons.

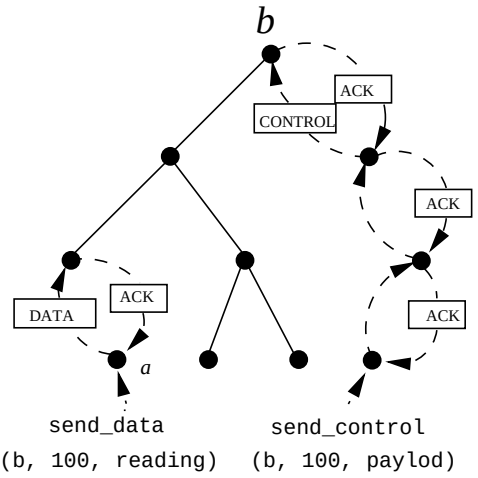


Fig. 2. Data and control packet send mechanisms.

Every connection in our protocol is represented by a base station address and a port number. To initiate the connection, the base station calls `listen` function provided by the transport layer interface to listen to some specific port. Any node willing to join the connection, calls `connect` with the base station address and corresponding port number (Figure 1). A connection table in each node keeps track of the established connections that the node is aware of. We emphasize that a node does not necessary participate in data collection for all connections stored in its connection table. Some nodes may simply serve as relay nodes. For each connection, the connection table keeps the corresponding basestation address, port number, and a flag to show whether the node itself is collecting data for that connection.

The `connect` message is sent towards the base station. It is processed at every hop and forwarded to the next hop. If an intermediate node is not aware of this connection, the base station address and port are added to the connection table with the participation flag indicating that the node only forwards messages in this connection. The `connect` message also contains the sender address and location, in addition to the basestation address and port number. The location is cached at every hop towards the base station. The reason behind forwarding the `connect` all the way to the base station is to cache the location of the joining node at all intermediate nodes. When received by the base station, an acknowledgment is sent back to the source. The connection is closed when the basestation calls a `close` function.

There are two different functions provided for sending data to the base station: `send_control` for sending control messages, and `send_data` for sending spatio-temporal data to the base station. As illustrated in Figure 2, `send_control` uses hop-by-hop acknowledgments to reliably transfer control packets to the destination (similar to [20], [23]). The structure of a control packet is depicted in Figure 3. A flag in the message header distinguishes this kind of messages from spatio-temporal data messages. The header also stores source and destination address and port number to distinguish between

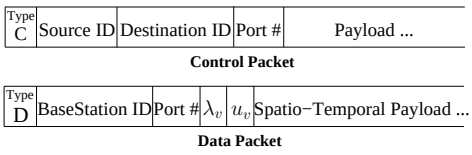


Fig. 3. Control and data packet structure.

different flows. When a packet is received from the network layer, a selective ACK message is sent back to the previous hop. The ACK message contains the port number and the sequence number of the acknowledged packet. If the packet is destined to the receiver, it is added to the receive buffer to be fetched by the receive function. Otherwise, it is forwarded towards the destination. This traditional approach supports any end-to-end communication (i.e. sensor-to-sink, sink-to-sensor, or sensor-to-sensor). Control messages form a small portion of the total traffic and are often critical. Therefore, we do not employ any congestion control mechanism for control packets.

The `send_data` function is responsible for transmitting the majority of the traffic and uses a specific format for its payload. This function is implemented differently from previous work, which is how the contributions of this paper are implemented. The payload contains an array of measurements (for example, temperature) and the associated node IDs. At the source, the sending application simply specifies its local node ID. However, as we explain later in detail, each measurement forwarded in the network may be attributed to more than one source node because of aggregation at the transport layer. Unlike the `send_control` function, spatio-temporal data may not be directly forwarded towards the base station depending on the next hop (Figure 2). If the next hop is participating in the same connection (i.e. running the same application), the `send_data` message is used to update the local array of measurements at the next hop and the message itself is dropped. Otherwise, the message is forwarded in the same way as the control packets until it reaches a participating node.

Formally, every node v, maintains $\bar{M}_v^v$ as an array of values of $\bar{m}_w^v$ for every node w in T_v (we drop the time variable to represent the latest value). Whenever a data message is received by v, the corresponding data points in $\bar{M}_v^v$ are updated and the message is dropped. The measurements are propagated at the next call to `send_data` at v where $\bar{M}_v^v$ is first updated using the local measurement at v given by the application. Assuming no congestion, it then sends the whole array of estimated data $\bar{m}_v^v$ to the parent and waits for an ACK message upon successful receipt of the packet by u. When an ACK message is received, v knows that the estimated values in u have been updated to $\bar{M}_v^v$ and the `send_data` function returns control to the application layer. The `receive` function at the base station simply returns the $\bar{M}_r^r$.

C. Congestion Control Mechanism

The main objective of the congestion control mechanism is to minimize the estimation error incurred at the base station while providing ensuring a weighted allocation of error among various applications running concurrently. A key idea here

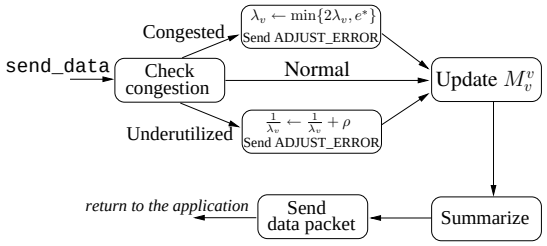


Fig. 4. Spatio-temporal data send mechanism.

is to control the error locally in a neighborhood so the global error is minimized. Let e_v denote the error between observations at a node and the values available to its parent. In traditional rate control approaches, the output flow of a node is directly controlled by its transfer rate. However, in our scheme, the output flow is indirectly controlled using e_v. The reason we use induced estimation error as a control mechanism rather than directly controlling the output flow is to manage the ultimate error contribution resulting from congestion control. For presentation simplicity, we first assume that all applications are equally important (i.e. have the same accuracy weight). Later in this section, we show how different weights are supported.

Our scheme assigns a value to each node v called *maximum tolerable error* and denoted by λ_v. The major principle of the protocol is to keep $e_v \leq \lambda_v$ while minimizing output data rate. Initially $\lambda_v = 0$ for all nodes. Upon detecting congestion, our protocol increases λ_v until congestion is eliminated. There are three components involved in this process (Figure 4): (i) The congestion detection component, (ii) Error control component which adapts λ_v to observed congestion conditions such that the congestion is eliminated, and (iii) The adaptive summarization component which finds the smallest summary for which e_v does not exceed λ_v, for any given value of λ_v. We first discuss the congestion control components, then present our summarization mechanism.

Congestion detection is widely studied in both wired and wireless networks. In traditional wired networks, the congestion can be easily detected by checking the output buffer size. The network is congested if the buffer is utilized more than a threshold. Wireless networks may need more sophisticated mechanisms such as identifying channel idle time (e.g. the mechanism proposed in CODA [24]). We treat the congestion detection component as a black box so any arbitrary mechanism can be used. At every call to the `send_data` function, our protocol first checks with the congestion detection component and determines the transport status at that node to be *congested*, *normal*, or *underutilized*. The current implementation simply uses thresholds on output buffer size.

After obtaining the status, the protocol increases or decreases λ_v in case of congestion or underutilization respectively. To find the appropriate estimation error which is just large enough to alleviate the congestion, we use a multiplicative (tolerance) increase, additive decrease method. When congestion is detected at node v, λ_v is increased to $2\lambda_v$. However, this increase does not always guarantee a decrease

in output rate. For example if $\lambda_v = 0$, the tolerance level would remain at 0. To solve this problem, the protocol first examines what would be a reasonable increase in λ_v to have an actual effect on the output rate. To do that, we consider the aggregate value currently being sent using the old value of λ_v. The protocol then computes the error introduced by adding one level of summarization. Let e^* be that error. Now, we adjust tolerance level using

$$\lambda_v(t+1) = \min\{2\lambda_v(t), e^*\} \quad (1)$$

In the case of underutilization, λ_v is modified as follows:

$$\frac{1}{\lambda_v(t+1)} = \frac{1}{\lambda_v(t)} + \rho \quad (2)$$

where ρ is a constant value protocol parameter. Now, one can claim that when the appropriate estimation error is found at the congested node v, we can locally use more aggressive summarization without making any changes in the protocol operation in other network parts. However, resources (energy and bandwidth) are being consumed by nodes in T_v to deliver error-free spatio-temporal data to v without knowing that this accuracy will be wasted. Although nodes in T_v may not be congested, their excessive resource usage will cause other nodes sharing the media or other applications running on those nodes to suffer. Therefore, the protocol informs all nodes in T_v about the change in the amount of tolerable error at node v and allows them to tolerate the same amount of error per sampling point. An special `ADJUST_ERROR` message is sent to all child nodes carrying λ_v. When receiving such a message, each node compares λ_v with its own tolerance level and picks the maximum.

In order to provide a differentiated service to different applications, each application i is given an accuracy weight α_i. Let E_r^i be the estimation error at the base station for the application i as defined in Definition 1. We design our protocol in such a way that for any two applications i and j we have:

$$\frac{E_r^i}{E_r^j} = \frac{\alpha_i}{\alpha_j}$$

This is simply achieved by modifying λ_v when running the congestion control scheme. Instead of λ_v, each node uses λ_v/α_i as the upper bound for e_v and follows the rest of the protocol as before. In Section IV, we show that this scheme converges to the desired weighted error distribution among various applications.

Internally to the transport layer, every packet contains the the base station address and port number (Figure 3) to identify each flow. The header also includes the value of λ_v at the time of transmitting packet where v is the transmitter. u_v is called the upper bound on the estimation error of the payload and defined as:

$$u_v = \max_{j \in T_u}\{|\bar{m}_j^v - m_j|\}$$

Finally, spatio-temporal payload is a summary of the array of estimated data $\bar{m}_v^v$.

D. Adaptive Summarization

In our scheme, adaptive summarization is the process of compressing the spatio-temporal payload to a smaller size while satisfying the maximum tolerable error bound computed above. This process is done at every call to the `send_data`. Let us denote this summery by $\bar{M}_v^u$ as this would be the values that u estimates for nodes in T_v. There are multiple degrees of summarization: a higher degree summarization means smaller payload but larger estimation error. Specifically for $n = |\bar{M}_v^v|$ samples, we define $\log(n)$ summaries as follows: We take the average of every consecutive pairs of values to obtain a set of $n/2$ samples. This creates the first summarization of $\bar{M}_v^v$. Continuing this process yields k-th summarization with the size of $\frac{n}{2^k}$. Next, we find the smallest summary which satisfies the following criteria:

$$(\bar{m}_v^u - m_v)^2 + \sum_{i=1}^{k} \sum_{j \in T_{w_i}} [(\bar{m}_j^u - \bar{m}_j^v)^2 + 2|\bar{m}_j^u - \bar{m}_j^v|u_{w_i}] \leq \lambda_v - \sum_{i=1}^{k} \lambda_{w_i} \quad (3)$$

To achieve this, we perform a binary search on $\log(n)$ different summaries. We observe that, for $\lambda_v = 0$ only 0-th summarization satisfies the condition (3). On the other hand, λ_v can be so large that even sending a single value does not violate (3). In this case, where spatial summarization no longer reduces the data rate, we start averaging values over time and sending one aggregate message instead of two. In particular, the protocol defers sending any data packet to the parent until the next call to `send_data` function while keeping the last values that u has received from v. When u does not receive any values from v, it uses the current values of $\bar{M}_v^u$.

Our scheme prefers spatial summarization over temporal summarization because unlike temporal summarization, the error introduced by spatial summarization can be calculated at the time of sampling. To perform bounded-error temporal aggregation, one needs to defer the summarization decision to after multiple sampling intervals. Also, observe that while more complex summarization could be used that introduces different degrees of temporal and spatial summarization to different data values, such complex techniques would require appropriately complex metadata to describe what summarization was performed. The advantage of our scheme lies in the simplicity of needed metadata as described next.

After summarization, λ_v and u_v are added to the packet header. In order to calculate u_v, v uses the u_{w_i} values reported in packet headers received from child nodes. In particular, v uses to the following to calculate u_v when composing each data packet header:

$$u_v = max_{j \in T_{w_i}, 1 \leq i \leq k}\{|\bar{m}_j^u - \bar{m}_j^v| + u_{w_i}\}$$

It can easily be verified that u_v is greater than $|\bar{m}_j^v - m_j|$ for any $j \in T_u$. After the data packet is composed, it is sent to the output buffer and the control return to the application.

IV. Analysis of Protocol Properties

Our proposed cyber-physical congestion control scheme uses the local estimation error at a subtree to control the data traffic. In this section, we first validate that this local decision is sufficient to control the error globally. We analytically show that our proposed scheme can indeed provide a controlled estimation error at the base station. Next, we study the computational complexity involved in the adaptive summarization process. We prove that the computational complexity is $O(\log(\log(n)))$, where n is the number of nodes. This allows the protocol to scale as the network grows larger. Finally, we prove that our approach successfully converges to a weighted error allocation, where each application i experiences an estimation error proportional to α_i when the protocol stabilizes.

A. Validation

The congestion control component in our protocol works based the fact that the estimation error at each node v is always less than or equal to λ_v. We first assume the single application scenario and show that the proposed scheme for adjusting summarization in a subtree respect the maximum tolerable error. Formally, we prove the following theorem:

Theorem 1: The estimation error contributed from any node v to its parent u is always less than or equal to λ_v.

Proof: We prove this theorem using induction on the height of the nodes. The leaf nodes only store their own measurement m_v. Thus, the error contributed to their parents are always less than λ_v as enforced by the protocol and the theorem is trivially true for the leaf nodes. Now, we assume that the data from child w_i introduces an error not exceeding λ_{w_i} to v. In other words:

$$\sum_{i\in T_{w_i}} (\bar{m}_i^v - m_i)^2 \leq \lambda_{w_i} \tag{4}$$

On the other hand, the error at data points in w_i sent to u can be expressed as follows:

$$\begin{aligned}\sum_{j\in T_{w_i}} (\bar{m}_j^u - m_j)^2 &\leq \sum_{j\in T_{w_i}} (|\bar{m}_j^u - \bar{m}_j^v| + |\bar{m}_j^v - m_j|)^2 \\ &\leq \sum_{j\in T_{w_i}} (\bar{m}_j^u - \bar{m}_j^v)^2 + \sum_{j\in T_{w_i}} (\bar{m}_j^v - m_j)^2 + \\ &\qquad 2\sum_{j\in T_{w_i}} |\bar{m}_j^u - \bar{m}_j^v||\bar{m}_j^v - m_j|\end{aligned}$$

Where the first inequality is based on $|a-b| \leq |a-c|+|c-b|$. By definition, $|\bar{m}_j^v - m_j| \leq u_{w_i}$. Using (4), we have:

$$\begin{aligned}\sum_{j\in T_{w_i}} (\bar{m}_j^u - m_j)^2 &\leq \sum_{j\in w_i} (\bar{m}_j^u - \bar{m}_j^v)^2 + \lambda_{w_i} + \\ &\qquad 2\sum_{j\in T_{w_i}} |\bar{m}_j^u - \bar{m}_j^v| u_{w_j}\end{aligned}$$

According to adaptive summarization (3):

$$\begin{aligned}&\sum_{j\in T_u} (\bar{m}_j^u - m_j)^2 \leq (\bar{m}_v^u - m_v)^2 + \\ &\sum_{i=1} k[\sum_{j\in Tw_i} (\bar{m}_j^u - \bar{m}_j^v)^2 + \lambda_{w_i} + 2\sum_{j\in T_{w_i}} |\bar{m}_j^u - \bar{m}_j^v| u_{w_j}] \leq \\ &\qquad \lambda_v - \sum_{i=1} k\lambda_{w_i} + \sum_{i=1} k\lambda_{w_i} = \lambda_v \end{aligned} \tag{5}$$

This completes the proof. ∎

From the above theorem we immediately conclude that the estimation error at the base station is bounded by λ_r. According to the proposed protocol, λ_r equals the maximum tolerable error at the most congested node in the network. Therefore, minimizing λ_v locally at the congested node simply minimizes the total estimation error bounded by λ_r.

We can generalize the argument to multiple applications with relative importance values of α_i. Similar to Theorem 1, we can show that the estimation error of application i contributed to node u by node v is bounded by λ_v/α_i. This can be proved just by replacing λ_v with λ_v/α_i. Therefore, the total estimation error in the multiple application case is bounded by λ_r/α_i.

B. Computational Complexity

There are several steps involved in sending a data packet. Although the sampling intervals of typical sensor networks are usually long enough to allow complex processing, we need to make sure that the protocol can easily scale when the network size increases. The following theorem describes the time complexity of our protocol:

Theorem 2: At every call to `send_data` function, our protocol performs $O(\log(\log(|T_v|)))$ summarizations.

Proof: First, note that the size of the complete sample array is $|T_v|$ at node v. Therefore, given λ_v, there are $\log(|T_v|)$ different summarizations to choose from. However, since the error of summaries are non-decreasing as the size decreases, we can use binary search to find the minimum size summary that satisfies our error constraint. Using binary search, the total number of summarizations is $O(\log(\log(|T_v|)))$ ∎

We should note that the summarization itself is a linear operation on the data packet. A same-order computation is performed in any transport protocol to send and receive the data packet itself ($O(|T_v|)$). So our protocol increases the total computational cost from $O(|T_v|)$ to $O(\log(\log(|T_v|))|T_v|)$ compared to previous approaches. Since the added term grows very slowly with n, we conclude that the protocol can scale well with increase in network size. On the other hand, the only parameter that affects the protocol convergence is ρ. Using a large value of ρ results in non-optimal solution while a small value will slow down our protocol. Ideally, ρ is inversely proportional to the number of applications.

C. Error Distribution

When running several applications on the same network, it is desirable to have resources distributed among them based

on their relative importance. Let λ_v^i be the maximum tolerable error computed by application i at node v. When using the cyber-physical congestion control scheme, all the applications using the network should incur estimation errors relative to their given α_i value. Formally, for every pair of applications i and j and any node v, with estimation errors of E_i and E_j, we should have:

$$\frac{E_i}{E_j} = \frac{\alpha_j}{\alpha_i} \tag{6}$$

We first show that the maximum tolerable error at each node converges to the same value for all applications as time passes.

Theorem 3: for every pair of applications i and j and any node v, we have $\lim_{t\to\inf} \frac{\lambda_v^i}{\lambda_v^j} = 1$

Proof:

According to the protocol we have:

$$\frac{1}{\lambda_v(t+1)} = \begin{cases} \frac{1}{\lambda_v(t)} + \rho & \text{non-congested} \\ \frac{1}{2\lambda_v(t)} & \text{congested} \end{cases} \tag{7}$$

When congestion occurs, we can write $\frac{\lambda_v^i}{\lambda_v^j}$ as follows:

$$\frac{\lambda_v^i(t+1)}{\lambda_v^j(t+1)} = \frac{\lambda_v^i(t)}{\lambda_v^j(t)} \tag{8}$$

On the other hand in non-congested operation, we have:

$$\frac{\lambda_v^i(t+1)}{\lambda_v^j(t+1)} = \frac{\frac{1}{\lambda_v^j(t)} + \rho}{\frac{1}{\lambda_v^i(t)} + \rho} = \frac{1+\rho\lambda_v^j(t)}{1+\rho\lambda_v^i(t)} \frac{\lambda_v^i(t)}{\lambda_v^j(t)} \tag{9}$$

Given (9) and for $\rho > 1$, we have $\frac{\lambda_v^i(t+1)}{\lambda_v^j(t+1)} \geq \frac{\lambda_v^i(t)}{\lambda_v^j(t)}$ if $\frac{\lambda_v^i(t)}{\lambda_v^j(t)} \leq 1$ and $\frac{\lambda_v^i(t+1)}{\lambda_v^j(t+1)} \leq \frac{\lambda_v^i(t)}{\lambda_v^j(t)}$ otherwise. Thus, we conclude that $\frac{\lambda_v^i}{\lambda_v^j}$ converges to 1 as time passes.

■

We conclude that for every pair of applications λ_r^i and λ_r^j converge to the same value as the protocol stabilizes. Since the estimation error of applications i and j converges to λ_r^i/α_i and λ_r^j/α_j respectively, we have:

$$\frac{E_i}{E_j} \to \frac{\lambda_r^i/\alpha_i}{\lambda_r^j/\alpha_j} = \frac{\alpha_j}{\alpha_i} \tag{10}$$

V. Evaluation

We evaluate our congestion control scheme using experiments on an indoor MicaZ mote [16] testbed. In particular, we study the estimation error under various network conditions. Unlike traditional network applications where the performance of a transport protocol is measured by throughput, estimation error is the major performance metric of a sensor network monitoring application.

First, in this section, we validate our protocols response to congestion. Next, we study the distribution of the induced estimation error among several applications when they share the network. Finally, we compare the performance of our protocol against two previous approaches to congestion control. We find ESRT [19] and RCRT [17] to be the best choice of protocols to compare against. The former is designed for monitoring applications without having packet-level reliability requirements while the latter is a flexible transport protocol with packet-level reliability and applicable to any sensor network. In both protocols, receiver is responsible for congestion control. We do not compare to our own prior work [1], because that work does not use congestion control.

A. Experimental Setup

We implemented our protocol in the LiteOS [14] operating system using C++. LiteOS offers a Unix-like thread-based interface for protocol implementation. The protocol API is provided to higher level applications as a C++ class. The transport layer uses a secondary thread to maintain receive buffer, send acknowledgments and perform forwarding. We run our experiments on 21 MicaZ motes deployed uniformly. In our implementation, the transport layer buffer is 500 bytes due to the limited memory available on MicaZ motes. Each mote is assigned a predefined one-byte ID. We use a set of fixed routing trees for different communication ranges. The range is represented proportionally to the distance to the nearest neighbor since the layout of sensors is a uniform triangular mesh.

We implement a light monitoring application in LiteOS, where the measured values are scalar. The application runs on each node asynchronously and sends the measurements every R seconds. R is called the sampling interval and is set to 5 seconds unless otherwise specified. We use the energy management module implemented for LiteOS [4]. However, we only use the radio operation energy cost which is normalized to one unit per transmitting one byte.

We implemented both RCRT [17] and ESRT [19] in LiteOS in order to perform comparison experiments. Both protocols use rate control for congestion management; however, they have different reliability semantics. No spatial aggregation is used in these protocols. The same application runs on top of each protocol to report light measurements. Unless otherwise specified, in order to stress all protocols, we use an average non-congestion-induced loss of 50% for all links to emulate an unreliable wireless medium.

For each experiment, the protocols run on the testbed for 10 minutes. Each experiment also runs 10 different times and the average is reported. Motes record their own sensor readings to the flash memory. The values returned by the receive function at the base-station are also recorded. Let $r_i(t)$ be the reading of mote i at sampling interval t recorded to its flash memory. Also, let $\hat{r}_i(t)$ be the value received by the base-station similarly preserved in its flash memory. At each run, the estimation error is obtained by computing the mean square error over time between recorded readings and the values from the base-station. This value is then divided by the total time (120 intervals, 5 seconds each) and the number of nodes (20) to calculate an average estimation error:

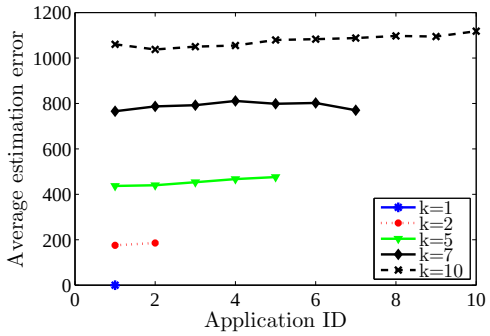


Fig. 5. Fairness of the adaptive aggregation congestion control ($\alpha_i = 1$).

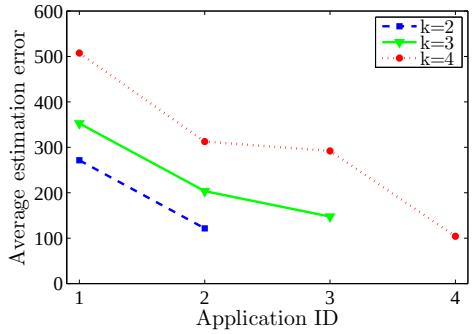


Fig. 6. Distribution of the estimation error among concurrent applications where $\alpha_i = i$.

$$\text{Estimation Error} = \frac{\sum_{t=1}^{120}\sum_{i=1}^{20}(r_i(t) - \hat{r}_i(t))^2}{20 \times 120}$$

Note that we present the absolute value of estimation error. Hence, it should be used as a relative measure for the purpose of comparison.

B. Protocol Validation

In the first experiment, we study how increasing the number of applications affects their performance, given a constant shared energy budget and equal weights ($\alpha_i = 1$) to all applications in Figure 5. In practice, different applications will focus on different sensor modalities. We start with one application using an energy budget of 20 bytes per second. In our system, it is more than enough for one application to transfer all the data without loss. Therefore, the resulting estimation error is zero.

As we increase the number of applications, k, a smaller share of the budget is secured for each. This leads to more estimation error as depicted in Figure 5. We observe that the estimation error is increased proportionally to the number of applications. Furthermore, the results verify that an increase in the error is equally shared among different applications as analytically predicted in Section IV.

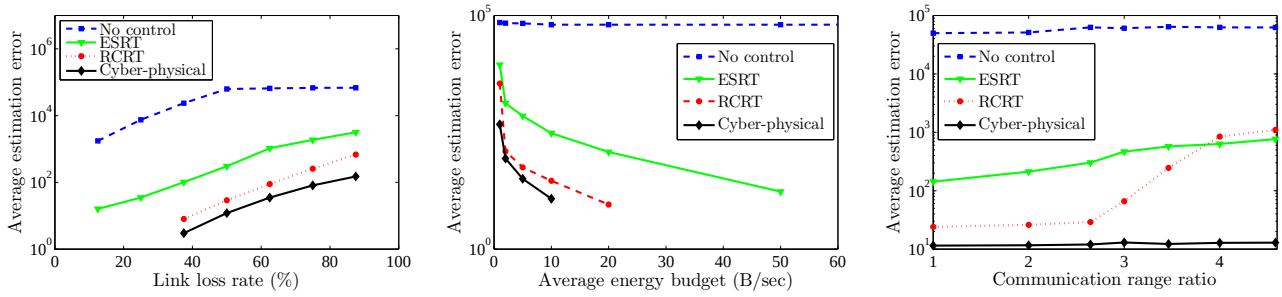
Next, we evaluate how proportional differentiated service is provided using different weights. We repeat the previous experiment with $\alpha_i = i$ and report the results in Figure 6. Based on the values of α_i we expect the first application to have twice the estimation error of the second application, three times of the third, and so on. Figure 6 shows that our protocol roughly maintains estimation errors corresponding to the given weight values.

C. Performance Comparison

Next, we compare performance of our protocol to others. In our first comparison, we plot the average estimation error when the wireless channel loss rate changes. The link loss rate is changed by modifying the physical radio interface code to deliberately drop packets. The rate varies between 0 and 87.5 percent and the estimation error is plotted in log scale in Figure 7(a). Recall that the energy budget for all nodes is 20 bytes per second. Considering this, none of the protocol incur any error with zero loss (i.e., fully reliable channels). Note that, the curves in Figure 7(a) do not include points with zero error since the y-axis is in log scale. One can conclude from the figure that our protocol outperforms RCRT and ESRT by 3 times when links become more lossy. In this experiment, the estimation error induced when using ESRT is not only because of data-agnostic congestion control but also because of the partial reliability it provides.

Next, we experiment with the effect of per node energy budget on the performance of these protocols. As explained in the experimental setup earlier, energy budget is represented by the number of bytes a node can transmit per second. Since the energy budget and channel bandwidth have the same effect on protocol performance and considering the fact that the energy budget can be easily controlled, we use it to study how protocols react to more resource constrained situations. We change the energy budget between 1 and 50 bytes per second and plot the average estimation error in Figure 7(b). The link loss rate is 50%. Again, y-axis is in log scale and as before, our protocol outperforms others in the low budget case. Observe that the estimation error remains constant when not applying any congestion control mechanism. This is because when links are very lossy, the high channel loss rate dominates the smaller error introduced by congestion.

In our final experiment, we evaluate the protocols against changes in communication range. Communication range is changed by changing the transmit power in the sensors. The minimum power considered as the unit range. In Figure 7(c) we show how increasing communication range affects the average estimation error. The communication range in this plot is normalized based on the unit range and presented as communication range ratio. As communication range increases, fewer hops are required to deliver a packet to its destination. This alleviates congestion. On the other hand, the larger interference caused by increased range reduces the effective bandwidth. This two-fold change causes the protocols to have different reactions. Our protocol, not suffering much from higher interference, keeps a steady performance while others show significant increase in the error as they encounter more collisions.



(a) (b) (c)

Fig. 7. Comparing the estimation error of ESRT, RCRT, and Cyber-physical congestion control for various: a) channel qualities, b) energy budgets, and c) communication ranges.

VI. Conclusion

We presented a novel congestion control mechanism for accurate estimation of spatio-temporal phenomena in wireless sensor networks performing monitoring applications. In our model, the system is reconstructing a physical phenomenon using sensor observations. The estimation error at the base station is the main performance metric of the wireless sensor network. Our protocol uses different levels of summarization to reduce data transfer rate while minimally impacting estimation error. To the best of our knowledge, this is the first work to employ adaptive aggregation as a congestion control mechanism that minimizes estimation error of physical phenomena. It is designed to easily scale with the size of the network with the minimal communication overhead. We implemented our protocol in LiteOS on MicaZ motes and evaluated its performance through testbed experiments. We showed that our protocol can promptly eliminate congestion. It can also provide weighted fairness among multiple applications. Finally, we showed that our approach can result in much more accurate estimation comparing to traditional rate-control mechanisms such as ESRT [19] and RCRT [17]. In particular, our protocol can transfer sensor data with 3 to 100 times less estimation error. This highlights that aggregation can be used as an effective way of controlling transfer rate in wireless sensor networks.

References

[1] H. Ahmadi and T. Abdelzaher. An adaptive-reliability cyber-physical transport protocol for spatio-temporal data. In *RTSS'09*, pages 238–247, Washington, DC, Dec. 2009.

[2] F. Bian, S. Rangwala, and R. Govindan. Quasi-static centralized rate allocation for sensor networks. In *4th IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON '07)*, pages 361–370, San Diego, CA, June 2007.

[3] M. Caccamo, L. Zhang, L. Sha, and G. Buttazzo. An implicit prioritized access protocol for wireless sensor networks. In *RTSS'02*, pages 39–48, Austin, TX, Dec. 2002.

[4] Q. Cao, D. Kassa, N. Pham, Y. Sarwar, and T. Abdelzaher. Virtual battery: An energy reserve abstraction for embedded sensor networks. In *RTSS'08*, pages 123–133, Barcelona, Spain, Dec. 2008.

[5] C. T. Ee and R. Bajcsy. Congestion control and fairness for many-to-one routing in sensor networks. In *SenSys'04*, pages 148–161, Baltimore, MD, Nov. 2004.

[6] E. Felemban, C.-G. Lee, and E. Ekici. Mmspeed: Multipath multi-speed protocol for qos guarantee of reliability and timeliness in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 5(6):738–754, 2006.

[7] V. C. Gungor, Özgür B. Akan, and I. F. Akyildiz. A real-time and reliable transport (rt) 2 protocol for wireless sensor and actor networks. *IEEE/ACM Transactions on Networking*, 16(2):359–370, 2008.

[8] B. Hull, K. Jamieson, and H. Balakrishnan. Mitigating congestion in wireless sensor networks. In *SenSys'04*, pages 134–147, Baltimore, MD, Nov. 2004.

[9] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *MobiCom'00*, pages 56–67, Boston, MA, Aug. 2000.

[10] Y. G. Iyer. STCP: A generic transport layer protocol for wireless sensor networks. In *IEEE Conference on Computer Communications and Networks (ICCCN'05)*, pages 449–454, San Diego, CA, Oct. 2005.

[11] K.-H. Kim, Y. Zhu, R. Sivakumar, and H.-Y. Hsieh. A receiver-centric transport protocol for mobile hosts with heterogeneous wireless interfaces. *Wireless Networking*, 11(4):363–382, 2005.

[12] S. Kim et al. Flush: A reliable bulk transport protocol for multihop wireless networks. In *SenSys'07*, pages 351–365, Sydney, Australia, Nov. 2007.

[13] A. Koubâa, A. Cunha, M. Alves, and E. Tovar. Tdbs: a time division beacon scheduling mechanism for zigbee cluster-tree wireless sensor networks. *Real-Time Syst.*, 40(3):321–354, 2008.

[14] LiteOS homepage. http://www.liteos.net/.

[15] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: a tiny aggregation service for ad-hoc sensor networks. *SIGOPS Operating Systems Review*, 36(SI):131–146, 2002.

[16] MicaZ Data Sheet. http://xbow.com/Products/Product_pdf_files/ Wireless_pdf/MICAz_Datasheet.pdf.

[17] J. Paek and R. Govindan. RCRT: rate-controlled reliable transport for wireless sensor networks. In *SenSys'07*, pages 305–319, Sydney, Australia, Nov. 2007.

[18] S. Rangwala, R. Gummadi, R. Govindan, and K. Psounis. Interference-aware fair rate control in wireless sensor networks. In *SIGCOMM '06*, pages 63–74, Pisa, Italy, 2006.

[19] Y. Sankarasubramaniam, Özgür B. Akan, and I. F. Akyildiz. ESRT: event-to-sink reliable transport in wireless sensor networks. In *MobiHoc'03*, pages 177–188, Annapolis, MD, June 2003.

[20] F. Stann and J. Heidemann. RMST: reliable data transport in sensor networks. In *1st IEEE International Workshop on Sensor Network Protocols and Applications (SNPA'03)*, pages 102–112, May 2003.

[21] K. Sundaresan, V. Anantharaman, H.-Y. Hsieh, and R. Sivakumar. ATP: A reliable transport protocol for ad hoc networks. *IEEE Transactions on Mobile Computing*, 4(6):588–603, 2005.

[22] R. Vedantham, R. Sivakumar, and S. Park. Sink-to-sensors congestion control. In *Proc. IEEE International Conference on Communications (ICC'05)*, volume 5, pages 3211–3217, Seoul, Korea, May 2005.

[23] C.-Y. Wan, A. T. Campbell, and L. Krishnamurthy. PSFQ: a reliable transport protocol for wireless sensor networks. In *1st Workshop on Wireless Sensor Networks and Applications*, pages 1–11, Atlanta, GA, 2002.

[24] C.-Y. Wan, S. B. Eisenman, and A. T. Campbell. Coda: congestion detection and avoidance in sensor networks. In *SenSys'03*, pages 266–279, Los Angeles, CA, Nov. 2003.

[25] Y. Zhou, M. Lyu, J. Liu, and H. Wang. PORT: a price-oriented reliable transport protocol for wireless sensor networks. In *16th IEEE International Symposium on Software Reliability Engineering (ISSRE'05)*, pages 117 – 126, Chicago, IL, Nov. 2005.