RT-SENMOS: Sink-Driven Congestion and Error Control for Sensor Networks

Charilaos Stais and George Xylomenos and Evangelos Zafeiratos
Mobile Multimedia Laboratory & Department of Informatics
School of Information Sciences and Technology
Athens University of Economics and Business
76 Patision, Athens 10434, Greece
E-mail: stais@aueb.gr, xgeorge@aueb.gr and zafiratosv@gmail.com

***Abstract*—A very common problem in the sensor networks area, where large numbers of sensors transmit data to a single sink node, is reliable transport. As sensors become cheaper and more powerful, sensor networks face the challenge of controlling all these devices in a efficient manner, which naturally leads to a sink-driven transport scheme. In this paper we propose RT-SENMOS, a sink-driven reliable transport protocol for sensor networks which places the responsibility for transmission rate allocation, congestion avoidance and error control to the sink. RT-SENMOS is intended to be integrated within a specific application which will set its operating parameters depending on its needs, operating on top of UDP/IP to avoid the need for kernel/superuser access. As RT-SENMOS is fully sink-controlled, it enables the use of simple and inexpensive fixed sensors, which offload all protocol intelligence to a more expensive but reusable sink. We present the design of the protocol, comparing it with a similar approach, called RCRT, and evaluate its performance using a real implementation.**

I. Introduction

A Wireless Sensor Network (WSN) may serve many applications and situations, from simple data collection and reporting, e.g. smart home monitoring or securing a controlled area, all the way to assisting human or robotic rescuers in disaster recovery situations. It is obvious that emergency cases are the most demanding, since the sensor network may be partially connected and the need for quick and reliable data transmission is paramount. The rescuer cannot count on sensor redundancy for gathering critical information, hence the sensor network must rely on a transport layer that can transfer data quickly, without losing packets due to congestion forming around the sink, where data from many sources naturally converge. As the network environment in WSNs in general, and in disaster recovery situations in particular, is very unpredictable, transport protocols should be quick to adapt to changes in prevailing conditions.

Initially motivated by the DIstributed Sensor systems For Emergency Response (DISFER) project [1], we designed the Reliable Transport protocol for SEnsor Networks with MObile Sinks (RT-SENMOS) [2], [3], a purely sink-controlled protocol, in the sense that the sink allocates transmission rates to all reachable sensors and manages the error recovery process depending on application objectives. A crucial aspect of our approach is that sensor handling depends on the application at hand, since sensors only need to implement a generic and very simple behavior, with the sink implementing the complex behavior required by that application. This allows the same sensors to be used for different application scenarios, by simply modifying the behavior of the sink. Furthermore, RT-SENMOS uses UDP/IP for packet transmission, therefore it can be fully embedded into the application, operating without kernel/superuser access.

In earlier versions of this work, we presented our sink-driven approach to congestion management and error handling [2] and a preliminary performance evaluation of a real RT-SENMOS implementation over an actual network [3]. In this paper, we present the latest version of RT-SENMOS, which incorporates a new error control mechanism, as well as a completely revised congestion control scheme that can handle sensors that only transmit at specific rates, for example, video cameras. We also present a detailed performance evaluation of a real RT-SENMOS implementation over an actual network and compare it against the Rate-Controlled Reliable Transport (RCRT) [4] protocol, another sink-controlled scheme for sensor networks. We apply the general design of RT-SENMOS to a disaster recovery scenario in a public building, in which different types of sensors attempt to communicate with a mobile rescuer that moves around the disaster area.

The remainder of this paper is organized as follows. In Section II we present our assumptions, motivate our design choices and show how RCRT compares to our work. In Section III we review protocol operation and we present the new features introduced to the scheme. Section IV describes our implementation and the experimental results. We conclude in Section V.

II. Rationale and Related Work

The communication environment assumed by RT-SENMOS is a multi-hop network, where neighboring nodes are connected via a broadcast wireless technology, such as WiFi, Bluetooth or ZigBee. Some of the nodes in the network (possibly all of them) host sensors which attempt to transmit their data to a destination node, the sink. The sink may be mobile, for example, a human or robot equipped with a computer. We assume that all nodes know the network address of the sink, that is, they are pre-programmed with a specific sink address which they periodically try to connect to. Sensors are relatively dumb, in the sense that they are not aware of a specific application scenario, they just try to send their data to the sink. On the other hand, the sink is customized for a specific application, employing the RT-SENMOS mechanisms to control the sensors as needed. Different applications can be used at different times over the same network, by simply taking over the pre-programmed sink address. Due to mobility and sensor outages, the sink may only be able to communicate with a subset of the sensors at any given time, therefore it must be able to adapt to rapid changes in the network environment. We assume that a dynamic routing protocol is used, ensuring connectivity between the sink and any reachable sensors [5]. Connectivity may be lost at any time though, due to node failures or sink mobility.

To allow RT-SENMOS to serve diverse applications, we assume multiple classes of sensors, which can be treated differently. In general, the sink first allocates bandwidth to each class and then it assigns a portion of that bandwidth to each sensor. Different algorithms can be used for each of these allocation schemes, depending on the application, although we generally expect congestion to be concentrated around the sink, as all transmissions converge there. Since RT-SENMOS is purely sink-controlled, the sensors are unaware of these policies. In our tests, sensors are divided into two classes: continuous and event sensors. A *continuous sensor* collects point data, e.g. a temperature value or a camera snapshot, and sends them periodically to the sink. An *event sensor* is triggered by an event, e.g. motion detection, to send some data, e.g. a 30-sec live video. We assume that event sensor data are delay sensitive, since they are correlated to a specific event in time. In contrast, we assume that continuous sensor data are not delay sensitive, as they are periodically refreshed.

III. Protocol Description

In this section we describe the behavior of the current version of RT-SENMOS. The detailed message formats and exchanges are omitted due to space limitations, as they are available in [3]. In general, the protocol operates in three phases: first, the sensor attempts to connect to the sink and send its class and its desired rate; then, one or more data exchanges take place, with recovery taking place either together with data transmission or in recovery rounds; finally, either side can drop the connection. Pairs of control messages that always follow each other are used to estimate the current *round-trip time* (RTT) between the sink and each sensor.

A. Error recovery

The basic error recovery mechanism of RT-SENMOS relies on the sink issuing Negative Acknowledgements (NAKs) for data packets and the sensors retransmitting the lost data. However, while most protocols retransmit lost data immediately, RT-SENMOS can differentiate its behavior based on sensor class. As the basic retransmission policy must also be implemented at the sensors, we have chosen two schemes which cover a wide range of applications, while allowing the sink to modify its behavior depending on the application.

For the continuous sensor class, and in general for any transmissions that are not delay-sensitive, retransmissions are performed in *recovery rounds*, as follows. First, all the data packets are transmitted, with the sink issuing NAKs. Next, the data packets for which NAKs were received are retransmitted, then the ones for which NAKs were received again are retransmitted, and so on, as in RMTPSI [6]. The sink can modify this policy by stopping the recovery process whenever it deems appropriate, for instance, when enough packets have been received to reconstruct the content. Furthermore, if a source loses connectivity with the sink before the transmission completes, the sink can approximately reconstruct the content, as it will have received incomplete data from beginning to end, rather than complete data only from the beginning.

For the event sensor class, and in general for any transmissions that are delay-sensitive, recovery has to be completed in a limited time frame. For this reason, packets are immediately retransmitted by the sensors when a NAK is received, as in RCRT [4]. The sink can modify this policy by choosing which packets to NAK, depending on their value to the application. For example, in MPEG video, packets containing I-frames are more valuable than those containing P-frames. In our experimental evaluation we have implemented a policy where the sink measures the recovery time for the past 10 retransmissions (a configurable parameter) to determine whether a retransmitted packet will arrive soon enough to be played out, based on the current position of the video player. If not, the packet is not NAKed at all.

B. Congestion management

The congestion management algorithm relies on two observations: first, congestion will most likely be concentrated around the sink, where all transmissions converge and, second, due to sink movement, the algorithm must converge quickly in order to be stable. For these reasons, while TCP and many other transport protocols use an Additive Increase - Multiplicative Decrease (AIMD) algorithm to probe the (unknown) available bandwidth on the end-to-end path, RT-SENMOS starts from the (known) available bandwidth around the sink.

Starting from the bandwidth available around the sink, RT-SENMOS first allocates a fixed portion of the bandwidth to each sensor class, depending on application requirements. Then, it performs rate allocation separately for each class, possibly using an entirely different algorithm per class. Note again that since RT-SENMOS is rate-controlled, the sensors are not aware of the actual scheme used, they just follow the sink's instructions. In our original scheme, each sensor announced its desired rate during connection establishment [3]; we have extended this to allow either indicating a single target rate, or a set of target rates. In our implementation, continuous sensors ask for a single target rate, while event sensors ask for a set of target rates, which could match different video qualities. The interpretation of the single target rate is that this is the minimum rate that the sensor would need to offer good service, so the rate allocation algorithm tries to match or exceed that rate allocation. The interpretation of the multiple target rates is that the sensor will use the highest one of these rates that is allowed by its rate allocation, meaning that ideally the rate allocation algorithm should exactly match one of these rates.

Whenever a new sensor is connected or an existing one is disconnected, the system is checked to see whether global adjustments need to be made, separately in each class. First, we calculate the total rates requested (not assigned) by the sensors of each class. If these are below the available bandwidth, new sensors will get their requested rate, existing sensors that were previously rate-limited will increase their rate by 20%, while all other existing sensors will get their requested rate. If, however, the requested bandwidth is higher than the available one, the available rate is shared *equally* among all sensors of that class; sensors with multiple rates are assigned the rate just below the equal share. New sensors are notified of their assigned rate in a start transmission message, while existing sensors are notified by a rate update message [3].

Congestion detection is based on the assumption that the congestion losses experienced by the sink in each connection are more akin to those occurring in a network with Random Early Detection (RED) [7]: due to the large number of sensors and the convergence of their data around the sink, congestion losses are expected to be randomly distributed among connections, with the loss probability increasing with the level of congestion. Furthermore, we assume that each sensor class has some minimum requirements in order to provide an acceptable service level, expressed by a maximum acceptable loss rate max_rate. The congestion management scheme monitors the current loss rate, $loss$, for each sensor, and uses the difference between max_rate and $loss$ to make its decisions. Specifically, when $max\_rate < loss$, the sink decreases the assigned rate in proportion to the current loss rate: $newRate = currentRate * (1 - loss)$. When $max\_rate > loss$, the sink increases the assigned rate in proportion to the difference between the threshold and the current loss rate: $newRate = currentRate * (1 + max\_rate - loss)$. As an example, if $max\_rate = 3\%$, when $loss = 1\%$ the rate will be increased by 2%, while when $loss = 5\%$ the rate will be decreased by 5%. To allow the network to react to these changes, we do not modify the rates for a small period after each change, unless if the sensor pool changes, in which case we apply the previous (global) algorithm. The event sensor rates are modified in the same way as described above, that is, we adjust their allocations to their actually desired rates.

IV. Performance evaluation

A. Experimental setup

To evaluate the performance of RT-SENMOS, we created a *mixed sensor* scenario with 14 event and 14 continuous sensors with an extra node acting as the sink. All sensors are connected to the sink over a single physical hop using a shared wireless channel. We used our own implementations of RT-SENMOS and RCRT written in Java, adding a loss injection module to emulate congestion losses at the sink. The emulated loss rate rather than being random, as in losses induced by wireless errors, was actually proportional to the current bandwidth usage, matching our assumption of a RED-like congestion loss model. Specifically, the loss rate at any given time was calculated by multiplying the fraction of the total bandwidth allocated by the sink with the target loss rate, which was 2%, thus ranging from zero to the target loss rate.

TABLE I: Experimentation parameters

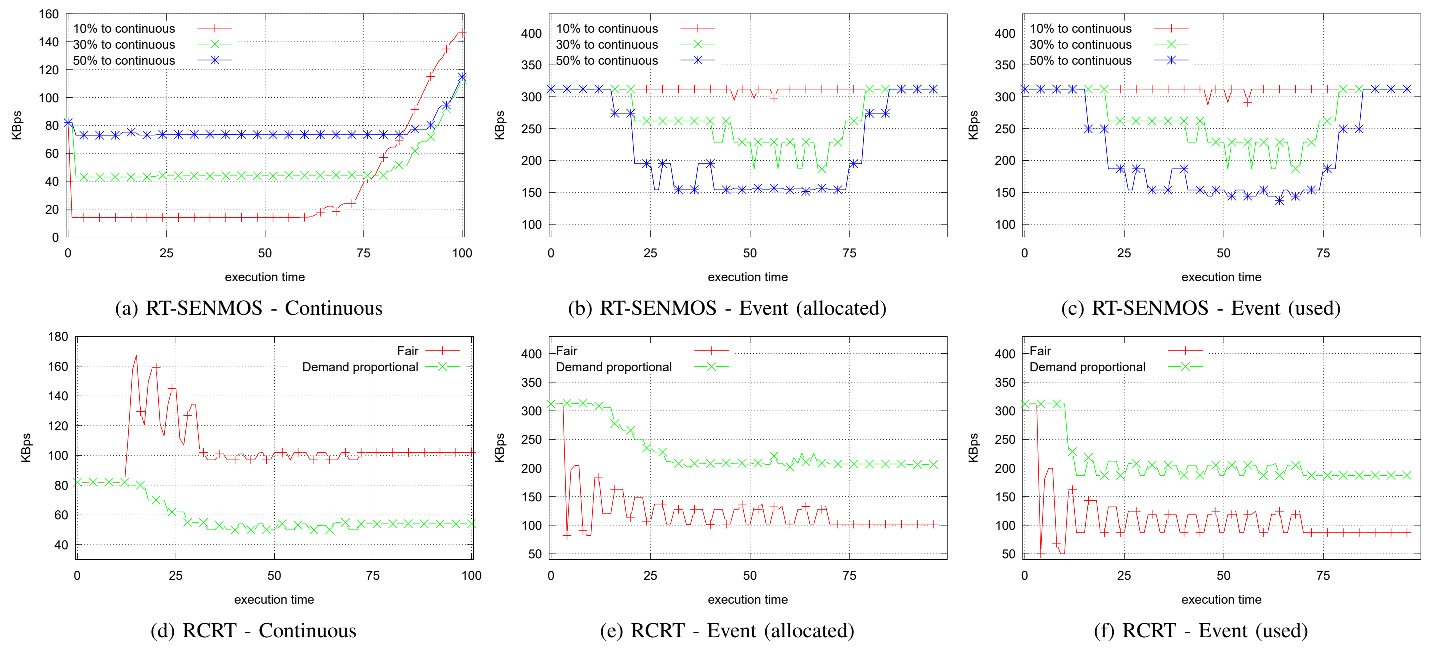
Parameter	Value
Continuous Sensor content size (MB)	8
Continuous Sensor bit rate (KBps)	82
Event Sensor content duration (sec)	30
Event Sensor bit rates (KBps)	50, 87, 187 and 312
Chunk size (bytes)	512
Loss samples stored per sensor	10
Bandwidth available at the sink (MBps)	2
Continuous Sensor share	10%, 30% and 50%
Target Loss rate	2%

We assumed that the continuous sensors were security cameras that sent individual 8 MB frames at a minimum desired rate of 82 KBps, while the event sensors when triggered sent 30 sec of live video at 50, 87, 187 or 312 KBps, depending on their rate allocation. The total bandwidth available at the sink was 16 Mbps (or, 2 MBps). In the test scenario we have a 100 m corridor, with event sensors deployed every 6.8 m; the rescuer connects to them while moving along the corridor. The continuous sensors are all connected at the beginning of the experiment and we wait until all transmissions are complete. The experimental parameters are listed in Table I. Regarding RCRT, we implemented its functionality, including all three rate allocation policies, on top of our message exchange scheme, using the same parameters values as in [4]. However, we only show results from the fair (all sensors get the same rate) and demand-proportional (each sensor gets the same proportion of its desired rate) policies, using the highest desired bit rate for event sensors, The rate-limited policy does not allow continuous sensors to exceed their minimum bandwidth, therefore it cannot be compared to our scheme.

B. Experimental results

In RT-SENMOS we assign a fixed fraction of the available bandwidth to each class, so Figures 1a and 1b show the mean bandwidth allocated to continuous and event sensors, respectively, when continuous sensors take up 10%, 30% and 50% of the bandwidth. The fraction allocated to each class balances their performance against each other, allowing the application to determine which sensor class it values more. We can also see that each sensor class quickly converges to a fair allocation of rates: event sensors are slowly connected, hence they only need to reduce their rates when a large number of them are in the pool, while continuous sensors are all connected at the beginning, hence they quickly reach their fair shares, which increase as some transmissions are completed.

Figures 1d and 1e show the same metrics for RCRT, using the two allocation policies available. With the fair policy, RCRT attempts to equally share the rate among all sensors, so they all converge around 100 KBps, which is more than enough for the continuous sensors, but too low for the event sensors. On the other hand, the demand-proportional policy attempts to allocate to each sensor class the same fraction of their desired rates, hence ending up with 50 KBps for continuous sensors (less than the desired 82 KBps) and 200 KBps for



(a) RT-SENMOS - Continuous (b) RT-SENMOS - Event (allocated) (c) RT-SENMOS - Event (used)

(d) RCRT - Continuous (e) RCRT - Event (allocated) (f) RCRT - Event (used)

Fig. 1: Mean Bandwidth Allocated/Used

event sensors (less than the highest rate, more than the second highest rate). As a result, even though the demand-proportional policy is better adjusted to a heterogeneous sensor mix, the allocations it produces are not satisfactory to either class.

While continuous sensors adapt to any rate allocation offered, meaning that the available and used bandwidths are generally the same, event sensors can only send at specific rates. As shown in Figure 1c, the mean bandwidth used by event sensors with RT-SENMOS is nearly exactly the same as the one allocated, shown in Figure 1b, while with RCRT the bandwidth used, shown in Figure 1f, is less than the one allocated, shown in Figure 1e. Specifically, with the fair policy the event sensors only use 87 KBps out of the 100 KBps allocated, while with the rate-proportional policy they only use 187 KBps out of the more than 200 KBps allocated. Finally, there is no appreciable difference in the experiment completion time between the different protocols and their variants.

V. Conclusion and Future work

We have presented and evaluated a reliable transport protocol for sensor networks, RT-SENMOS, especially suitable for disaster recovery applications with a rescuer as the sink. RT-SENMOS is purely sink driven and implemented at the application layer, thus allowing application policies to be set at the sink, using a basic toolkit of error and congestion control policies that we have implemented. This allows the sink to split the bandwidth between different classes of sensors and within each sensor class depending on application preferences, without modifying the sensors. RT-SENMOS controls each sensor individually, depending on the level of congestion, which is estimated by measuring packet loss. We have also provided a performance evaluation of an actual implementation of our protocol over a real network with emulated losses against RCRT. The results show that our approach exploits available bandwidth better than RCRT, as it tries to meet the actual requirements of each class. The experiments show that the gain from our approach is more visible in scenarios with sensors that can operate using a set of fixed rates rather than any rate.

Acknowledgment

This research was financed by the Research Centre of the Athens University of Economics and Business, in the framework of the project entitled “Original Scientific Publications”.

References

[1] “Distributed sensor systems for emergency response (DISFER) project,” http://www.aueb.gr/disfer, accessed: 2016-07-29.

[2] C. Stais, G. Xylomenos, and G. F. Marias, “Sink controlled reliable transport for disaster recovery,” in *Proc. of the International Conference on Pervasive Technologies Related to Assistive Environments*, 2014.

[3] C. Stais and G. Xylomenos, “RT-SENMOS: Reliable transport for sensor networks with mobile sinks,” in *Proc. of the IEEE Symposium on Computers and Communication (ISCC)*, 2015, pp. 105–110.

[4] J. Paek and R. Govindan, “RCRT: Rate-controlled reliable transport for wireless sensor networks,” in *Proc. of SenSys ’07*, 2007, pp. 305–319.

[5] K. Karenos and V. Kalogeraki, “Traffic management in sensor networks with a mobile sink,” *IEEE Trans. on Parallel and Distributed Systems*, vol. 21, no. 10, pp. 1515–1530, 2010.

[6] C. Stais, G. Xylomenos, and A. Voulimeneas, “A reliable multicast transport protocol for information-centric networks,” *Journal of Network and Computer Applications*, vol. 50, pp. 92 – 100, 2015.

[7] S. Floyd and V. Jacobson, “Random early detection gateways for congestion avoidance,” *IEEE/ACM Transactions on Networking*, vol. 1, pp. 397–413, 1993.