Hop-by-Hop Congestion Control and Load Balancing in Wireless Sensor Networks

Can Basaran, Kyoung-Don Kang, Mehmet H. Suzer
Department of Computer Science
State University of New York at Binghamton
{*cbasaran,kang,msuzer*}*@cs.binghamton.edu*

***Abstract*—Due to the relatively high node density and source-to-sink communication pattern, wireless sensor networks (WSNs) are subject to congestion and packet losses. Further, the availability of low-cost hardware, such as Cyclops cameras, is promoting wireless multimedia sensing to support, for example, visual surveillance. As a result, congestion control is becoming more critical in WSNs. In this paper, we present a lightweight distributed congestion control method in WSNs. We develop new metrics to detect congestion in each node by considering the queue lengths and channel conditions observed in the one-hop neighborhood. Based on the estimated level of congestion, each node dynamically adapts its packet transmission rate and balance the load among the one-hop neighbors to avoid creating congestion and bottleneck nodes. In a simulation study performed in OMNeT++, our approach significantly enhances the end-to-end (e2e) packet delivery ratio and reduces the e2e delay without increasing the total energy consumption compared to the tested baseline approach.**

Keywords: Congestion Avoidance, Load Balancing, Fuzzy Logic Control

I. Introduction

Wireless sensor networks (WSNs) have a number of applications such as environmental monitoring, target tracking, disaster recovery, and surveillance. A WSN typically consist of one or more sink nodes and a large number of sensor/relay nodes. As the research on WSNs continues to evolve, low-end cameras are being adapted to extend the capability of sensing. For example, Cyclops cameras [1] support up to 352×288 image resolution and 10 frames per second (fps) using low-power CMOS imaging technology. They are developed for resource-constrained sensor nodes to support a number of important applications such as visual surveillance and habitat monitoring [2].

In general, the relatively high node density, source-to-sink communication patterns, and multi-hop network topology in typical WSNs turn congestion control into a challenging problem even at moderate sampling rates. An accurate offline estimation of the feasible network bandwidth is very difficult to derive in application domains where a random deployment, for example, by an aircraft is expected. Moreover, variations in channel capacity, link quality, and neighbor density make the estimation very challenging even for manually deployed topologies [3]. It becomes more challenging to control congestion in WSNs as more advanced sensors, such as CMOS cameras and microphones, are added to WSNs. Thus, in this paper, we focus on estimating the degree of congestion and avoiding congestion based on the estimation in a WSN.

Our cross-layer approach, called CONSEQ (CONtrol of SEnsor Queues), aims to reduce congestion (if any) and, thereby, decrease the delay and energy consumption due to packet losses and retransmissions in WSNs. The contribution of our work is twofold: We develop (1) new metrics to estimate the degree of potential congestion in WSNs and (2) an efficient congestion control scheme using the estimation metrics. The design objective of CONSEQ is to support lightweight congestion control integrated with load balancing in WSNs. CONSEQ works on the per-hop basis; each node performs localized control within its one-hop neighborhood in a scalable manner. CONSEQ require no multi-hop or e2e feedback from the sink(s) to the sources. Instead, a node in a WSN only observes its one-hop neighbors using our metrics for congestion detection. As CONSEQ is executed in a hop-by-hop manner, it provides quick reaction to potential congestion, while incurring little overhead.

A. Metrics for Congestion Detection

In a wired network, congestion is detected based on the round trip time (RTT) or queue length in a router. However, these metrics are not appropriate for WSNs. The RTT between a source and sink could be long due to relatively low-rate, lossy multi-hop wireless communications. As a result, a reaction to congestion based on the e2e RTT could be too late and ineffective. Also, each node in WSNs could be both a source and router; therefore, a node has to consider not only its own load but also the potential congestion in its vicinity indicated by, for example, the neighboring nodes' queue sizes.

Ideally, the estimation of congestion is desired to be localized in the one-hop neighborhood without requiring potentially expensive e2e or multi-hop measurement. To this end, we first define the notion of the *virtual queue length* for an arbitrary node i ($1 \leq i \leq N$) in a WSN that consists of $N(\geq 1)$ nodes. Suppose that node j belongs to node i's forwarding set (FS); that is, node j has a smaller hop-count to the sink than node i does. Upon successfully receiving a packet from node i, node j piggybacks its queue length information to the ACK packet sent to node i. Node i computes node j's virtual queue length by adding its own queue length, node j's queue length, and the total number of packet drops observed since the last successful transmission from nodes i to j.

In this way, node i estimates the hypothetical worst case queue length of node j in the future where node i transmits all its packets to node j, while considering the observed wireless link condition between nodes i and j. Node i probabilistically balances the load among the nodes in the FS based on the virtual queue lengths. Let us suppose that node k also belongs to node i's FS and it has a longer queue than node j does. In this case, node i probabilistically forwards more packets to node k. If node i's queue length grows/shrinks, load balancing becomes less/more sensitive to the difference of node j's and node k's queue lengths.

The worst case estimation based on the virtual queue lengths could be too pessimistic especially in the presence of load balancing. To address this issue, we develop another metric called *effective queue length (EQL)*. (A detailed description of EQL is given in Section IV.) After computing the virtual queue length for each node in its FS, node i computes EQL_i to estimate its expected impact on the potential congestion in the FS by considering the effect of load balancing within the FS.

B. Hop-by-Hop Congestion Control

In CONSEQ, each node manages its EQL by dynamically adapting the rate of packet transmissions to the nodes in its FS, while balancing loads among them, if necessary, to avoid creating congestion and bottleneck nodes in the FS. Instead of relying on a relatively slow and expensive e2e feedback from the sink to the sources, each node tries to ensure that its EQL does not significantly exceed the specified threshold, called EQL set-point. In this way, our protocol aims to avoid severe congestion and load imbalance in a WSN that may significantly degrade the performance and energy efficiency.

To closely support the EQL set-point, we apply *fuzzy control theory* [4] that is very effective to manage the performance of a highly dynamic nonlinear system, e.g., a WSN, without requiring a mathematical model of the system. A fuzzy controller is essentially a direct (nonlinear) mapping between its input, e.g., EQL error = EQL set-point $-$ current EQL in a node, and output, e.g., the required rate adjustment for the node to achieve the EQL set-point, unlike other controllers such as PID (proportional, integral, and derivative) controllers [4]. Fuzzy control theory [4] provides formal techniques to represent, manipulate, and implement human experts' heuristic knowledge for controlling a plant, e.g., a wireless network, via if-then rules rather than relying on mathematical modeling of the plant. Thus, it can effectively handle real world uncertainties in a highly dynamic nonlinear systems such as WSNs.

Notably, our fuzzy controller operates in an *event-driven* manner in contrast to most of the existing feedback controllers [5]–[7] invoked at every sampling period. A controller in node i is invoked upon the arrivals of a specified number of packets at node i. By taking an event-driven approach, we ensure that our controller reacts quickly to a large number of packet arrivals in a short time interval, while saving energy by avoiding unnecessary control algorithm executions when few packets arrive. To the best of our knowledge, no prior work has applied event-driven fuzzy control techniques to manage the effective queue length for hop-by-hop congestion control in WSNs.

C. Performance Evaluation

For performance analysis, we have done an extensive simulation study in OMNeT++ [8]. We compare the performance of CONSEQ to Priority-based Congestion Control Protocol (PCCP) [9]. We select PCCP as the baseline, because it is specially designed for efficient hop-by-hop congestion control in multimedia WSNs as discussed before. In the simulation study, CONSEQ considerably outperforms PCCP. It substantially increases the delivery ratio, while reducing the energy usage and e2e delay compared to PCCP. More specifically, CONSEQ enhances the e2e delivery ratio by up to approximately four times, and decreases the e2e delay by up to five times. Also, by reducing the numbers of packet drops and retransmissions, CONSEQ reduces the energy consumed by each packet successfully delivered to the sink by 50% on average.

The rest of the paper is organized as follows. Related work is discussed in Section II. Section III describes the overall structure of CONSEQ. The congestion estimation metrics and load balancing scheme are discussed in Section IV. The rate control scheme for congestion avoidance is discussed in Section V. Performance evaluation is given in Section VI. Finally, Section VII concludes the paper and discusses future work.

II. Related Work

Increasing research in multimedia sensor networks [1], [2], [10] is altering the perception of sensor networks. A conventional view of WSNs generally suggests low-end sensor nodes providing a few bytes of data with a few readings per second at most. On the other hand, a newly establishing view embraces large sensor data and much more frequent sensor readings. For example, a sensor node equipped with a Cyclops camera needs to transmit hundreds of kilobytes of data per second [11].

Adaptive Rate Control (ARC) [12] is a rate control scheme tightly integrated with the MAC layer. It has MAC layer extensions to support fairness and energy efficiency by changing the MAC protocol based on the traffic load. The proposed rate control mechanism is based on the additive increase multiplicative decrease (AIMD) policy [13] in which overhearing a successful packet forwarding triggers an increase in the injection rate. Other approaches to rate control in WSNs, such as CODA [14], IFRC [15], and RCRT [16], rely on AIMD too. An advantage of AIMD is that it does not require any prior knowledge about the link capacity. However, its long convergence time to the achievable rate is the drawback. Rather than applying AIMD, CONSEQ adapts the rate via the neighborhood feedback based on fuzzy control theory.

CODA [14] detects congestion by measuring the queue length and estimating the channel load by periodically listening to the channel. If congestion is detected, back-pressure is

propagated downstream towards the source(s). Also, a source can require the sink to provide an e2e feedback, if the event rate exceeds a certain threshold. CONSEQ does not require expensive channel listening to detect congestion. Neither does it require explicit e2e or multi-hop feedback.

Event-to-sink reliable transport (ESRT) [17] is a reliable transport protocol that dynamically adapts the report rate of sensor nodes in order to support congestion control and enhance the reliability in an energy-conscious fashion. As the ESRT algorithm is mainly implemented in the sink, the response to congestion via rate adaptation at the source(s) could be relatively slow.

CCF [18] aims to support congestion control and fairness in WSNs. In CCF, a node measures the available downstream bandwidth in terms of service time and divides it among its down stream nodes. A node allocates its bandwidth to a child node in proportion to the size of the subtree rooted by the child node.

MCCP [19] detects congestion by measuring the buffer size and packet delivery time between transport layers of two nodes. MCCP creates a transport layer TDMA schedule for rate assignment and in-order delivery of event packets to the underlying layers. However, TDMA scheduling requires time synchronization between nodes, incurring non-trivial complexity and overheads.

The WRCP [20] is designed to support quick convergence than the existing approaches based on AIMD, such as [12], [14]–[16]. However, unlike CONSEQ, WRCP provides e2e feedback to the sources for congestion control. Also, congestion information is collected periodically. Thus, the congestion control method can be executed more/less frequently than necessary when the traffic intensity is low/high. In contrast, CONSEQ requires no e2e feedback. It also employs an event-driven approach to congestion detection.

To detect congestion, PCCP [9] measures the ratio of the packet inter-arrival time to the packet service time. PCCP is unique in that it supports hop-by-hop congestion control requiring no e2e feedback from the sink to sources. It also employs event-driven congestion detection. In this paper, we select PCCP as the baseline for performance comparisons, as it is closest to CONSEQ in terms of the design philosophy for hop-by-hop congestion control.

III. System Overview

The architecture of CONSEQ is shown in Figure 1 and its overall behavior for congestion control is described in this section.

In this paper, we assume that sensor nodes are stationary and only a fraction of nodes in the network are source nodes equipped with cameras to support, for example, surveillance. The rest of the nodes forward data toward the sink. Also, we assume that each node employs CONSEQ to estimate the degree of congestion in its one-hop FS and accordingly control its rate of scheduling packet transmissions to the FS and adapt load balancing decisions, if necessary, to avoid congestion and creating bottleneck nodes in a WSN. A node decreases the scheduling rate, if the congestion increases and vice versa. The degree of scheduling rate adaptation is derived via fuzzy control.

In CONSEQ, a node with more than one neighbors in its FS supports forwards each packet to one of the nodes in the FS to support load balancing. The forwarding set is constructed by flooding initiated at the sink node. A flooded hello packet contains a hop-count field initialized to 1 by the sink. The hop-count of an arbitrary node in the WSN except the sink is initialized to infinity. A node that broadcasts a hello packet with a smaller hop-count than a receiving node's hop count is added to the receiving node's FS. Accordingly, the receiving node updates its one-hop neighbor table, i.e., NB-Table in Figure 1. The receiving node determines its distance to the sink by taking the minimum hop-count encountered. After that, the receiving node rebroadcasts the message with an incremented hop-count.

CONSEQ is a cross layer protocol implemented in the transport and network layers with a few MAC layer modifications as shown in Figure 1. A description of the control flow in an arbitrary node that is part of a WSN with $N(> 1)$ nodes follows:

1) Each node i $(1 \leq i \leq N)$ computes the virtual queue length for each node j that belongs to its forwarding set FS_i. Based on the virtual queue lengths, the load balancer in node i probabilistically distributes load among the nodes in the FS, while computing node i's effective queue length, EQL_i.
2) Upon receiving a specified number of ACK packets, node i's rate adapter in Figure 1 generates the packet scheduling rate control signal based on EQL_i by applying fuzzy control techniques [4].
3) As shown in Figure 1, this control signal is forwarded to both the scheduler and the application layer. The scheduler is invoked periodically to schedule the next packet transmission. By increasing the scheduling period, CONSEQ in node i decreases the packet injection rate to the MAC layer, if necessary, to avoid congestion in FS_i. On the other hand, it increases the packet injection rate, if necessary, to utilize the available channel capacity more efficiently. Similarly, the application adapts the

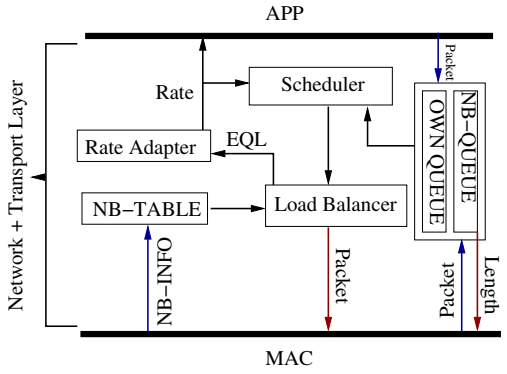

Fig. 1. CONSEQ Architecture

data generation rate according to the control signal, if necessary, to avoid congestion in FS_i. Packets generated by node i and through-traffic packets are buffered in the own queue and neighbor (NB-Queue) in node i, respectively.

4) At every packet injection period, the scheduler in node i selects the next packet to be forwarded towards the sink in a round robin manner based on the original source addresses of the packets buffered at node i.[1]
5) The load balancer in node i selects a specific node in FS_i to which the packet selected in the previous step will be forwarded. Especially, it selects a node in FS_i to avoid generating a hot-spot node, which receives an excessively large number of packets compared to the other nodes in the FS.
6) If node j ($\in FS_i$) successfully receives a packet from node i, node j increments its NB-Queue length and returns an ACK packet to node i. The ACK packet includes node j's NB-Queue length to help node i compute node j's congestion level.

IV. Load Balancing and Congestion Estimation

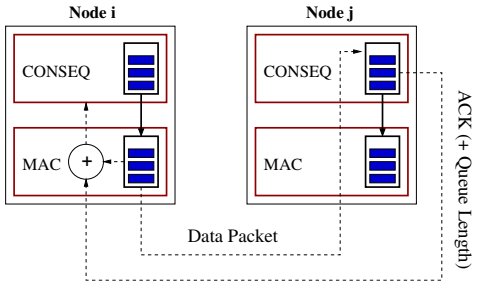

Fig. 2. Packet Transmission and ACK Delivery

Figure 2 shows the data communication and congestion information flow between a pair of one-hop neighbors, nodes i and j. A packet is dequeued from the network layer queue and inserted into the MAC layer queue of node i. If the packet is transmitted successfully to node j, it is inserted into node j's NB-Queue.

In CONSEQ, an arbitrary node i in a WSN makes probabilistic load balancing decisions among the nodes in its FS_i based on their virtual queue lengths. More specifically, node i computes the virtual queue length q_j of a node $j \in FS_i$ according to the following equation:

$$q_j = MAC_i + NQ_j + \beta \times D_i \tag{1}$$

where MAC_i is the number of packets in node i's MAC layer queue, NQ_j is the number of the packets in node j's NB-Queue, D_i is the number of packets dropped by n_i due to an excessive number of retransmissions after the most recent successful transmission, and β is the limit for retransmitting a single packet. (In our performance evaluation discussed in Section VI, $\beta = 4$.) In this way, we avoid generating false perception of the queue length reduction when in reality packets are dropped at the underlying MAC or physical layer.

Using the virtual queue length q_j, we compute the weight for node $j \in FS_i$:

$$w_j = \alpha^{q_j} \tag{2}$$

where $0 < \alpha < 1$. Therefore, w_j becomes smaller as q_j increases. As $\alpha < 1$, node selection in the FS is less sensitive to small differences between virtual queue lengths of the nodes in FS_i compared to making $\alpha \geq 1$. (In Section VI, $\alpha = 0.9$.)

After computing the weight for every node in FS_i, node i selects and forwards a packet to node $j \in FS_i$ with the probability P_j computed as follows:

$$P_j = \frac{w_j}{\sum_{j=1}^{N_i} w_j} \tag{3}$$

where N_i is the cardinality of FS_i. Hence, each node in a WSN probabilistically balances loads among the nodes in its FS based on their virtual queue lengths.

Finally, node i computes its effective queue length as follows:

$$EQL_i = \sum_{j=1}^{N_i} (P_j \times q_j) \tag{4}$$

By computing EQL_i based on the virtual queue lengths, observed packet drops, and the effect of probabilistic load balancing, node i estimates the degree of congestion in FS_i. Further, each node uses its EQL for dynamic rate adaptation to avoid congestion as discussed next.

V. Dynamic Rate Adaptation via Fuzzy Control

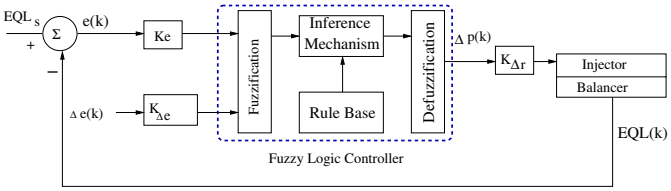

Fig. 3. Fuzzy Packet Injection Rate Controller in Each Node

In each node, the rate adapter dynamically adjusts the rate of packet injection to the underlying MAC layer, if necessary, to avoid congestion in the FS. Figure 3 shows the fuzzy rate controller implemented in an arbitrary node i in a WSN. As CONSEQ is employed at every node, we drop the index term, i, in the remainder of this section.

Our approach to injection rate control for congestion avoidance consists of the following steps:

1) The rate adapter in Figure 1 computes the error, $e(k)$, and change-in-error, $\Delta e(k)$, in terms of rate control using Eq. 5 and Eq. 6 to be discussed in Section V-A.
2) The fuzzification interface in Figure 3 fuzzifies $e(k)$ and $\Delta e(k)$ to convert $e(k)$ and $\Delta e(k)$ to the corresponding

[1]CONSEQ is not tied to a specific scheduling algorithm. A more advanced scheduling algorithm, such as the weighted fair queuing method [21], can be used to further improve the performance at the cost for more complex scheduling and corresponding overheads. A thorough investigation of potential trade-offs is reserved for future work.

linguistic error and change-in-error values such as negative small (NS) or positive large (PL).

3) The inference mechanism in Figure 3 looks up the rule-base to select the if-then rules corresponding to the linguistic error and change-in-error values.
4) The defuzzification interface in Figure 3 derives a crisp control signal $\Delta p(k)$ expressed as a real number from the selected if-then rules.
5) Using the crisp control signal, the period of packet scheduler invocation is adapted, if necessary, to avoid congestion by ensuring that the EQL in a node does not substantially exceed the desired EQL set-point, EQL_s in Figure 3. For example, if $\Delta p(k) = 0.1$, the packet scheduler invocation period is extended by 0.1s to avoid congestion in the node's FS by closely supporting EQL_s.

A more detailed description of these steps is given in the following subsections.

A. Computing the Error and Change-in-Error

CONSEQ is invoked as soon as a node receives a specified number, $N_a(\geq 1)$, of ACK packets for the data packets it forwarded to the nodes in its FS. The node computes its EQL as discussed before. Further, the rate adaptor of the node computes the error as follows:

$$e(k) = (EQL_s - EQL(k))/\Delta t \tag{5}$$

where k stands for the total number of times the controller has been invoked so far and Δt is the time elapsed between the first and last of the N_a ACK packet receptions.

In addition to the error, the rate adaptor computes the change in error $\Delta e(k)$:

$$\Delta e(k) = e(k) - e(k-1) \tag{6}$$

to observe potential network status changes in the neighborhood.

B. Fuzzification using Linguistic Variables and Membership Functions

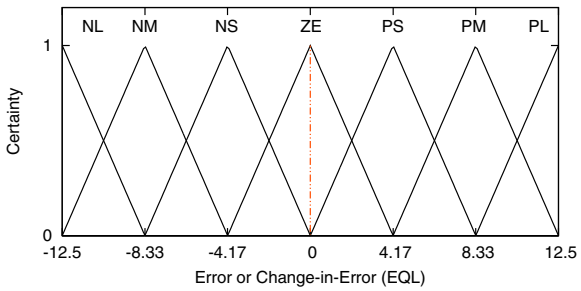

Fig. 4. Input Membership Functions for Error and Change-in-Error (NL: Negative Large, NM: Negative Medium, NS: Negative Small, ZE: Zero, PS: Positive Small, PM: Positive Medium, PL: Positive Large)

At the k^{th} invocation, the fuzzification interface in Figure 3 converts $e(k)$ and $\Delta e(k)$ to the corresponding linguistic variables such as negative small (NS). This conversion is based on fuzzy set theory to deal with uncertainties and imprecise information prevalent in the real world [4]. Notably, fuzzy set membership is not binary but continuous to avoid the sharp boundary problem of a binary threshold.

Figure 4 shows the fuzzy membership functions used in this paper to convert $e(k)$ and $\Delta e(k)$ to corresponding linguistic variables. For example, if $e(k) = -4.17$, the linguistic variable *error* produced by fuzzifying $e(k)$ belongs to NS in Figure 4 with certainty 1. As another example, if $\Delta e(k) = 1.0425$, the linguistic variable *change-in-error* belongs to ZE with certainty 0.75 and PS with certainty 0.25, respectively.

In this paper, we set maximum $EQL = 25$ and $EQL_s = 12.5$ packets. Therefore, $e(k)$ is bounded between -12.5 and 12.5 as shown in Figure 4. We also bound $\Delta e(k)$ to the range of [-12.5, 1.25]. If $\Delta e(k) > 12.5$ (or $\Delta e(k) < -12.5$), it is truncated to 12.5 (or -12.5) to support the bounded input to (and output from) the fuzzy controller.

		$\Delta e(k)$						
		NL	**NM**	**NS**	**ZE**	**PS**	**PM**	**PL**
e(k)	**NL**	PL	PL	PL	PL	PM	PS	ZE
	NM	PL	PL	PL	PM	PS	ZE	NS
	NS	PL	PL	PM	PS	ZE	NS	NM
	ZE	PL	PM	PS	ZE	NS	NM	NL
	PS	PM	PS	ZE	NS	NM	NL	NL
	PM	PS	ZE	NS	NM	NL	NL	NL
	PL	ZE	NS	NM	NL	NL	NL	NL

TABLE I
FUZZY RULE-BASE FOR CONGESTION CONTROL

C. Rule-Base Design

Rule-base design is an important step to develop an effective fuzzy controller. In this paper, we design the rule-base in Table I by considering the relations between the error and change-in-error. In particular, the linguistic control signals for packet scheduling period adaptation in Table I are derived from the following linguistic equation:

$$\text{control signal} = -(\text{error} + \text{change-in-error}) \tag{7}$$

The key idea behind Eq. 7 is to derive the linguistic control signal(s), i.e., the consequent(s) of the selected if-then rules based on the *error and the degree of self-error-correction by the node itself* indicated by the relation between the error and change-in-error. More specifically, we consider the absolute magnitudes and signs of the error and change-in-error. Even if the error is (positive or negative) large, we consider the node is effectively correcting the error for itself, if the change-in-error is also large and has the opposite sign. Otherwise, using Eq. 7, we generate a linguistic control signal to compensate for the potential insufficiency or excessiveness of self-correction in terms of EQL management. For example, if the error is NL and change-in-error is NL, the EQL of the node is longer than EQL_s and it is increasing fast; therefore, the corresponding consequent of the if-then rule is PL in Table I. As a result, the packet injection period is largely increased to reduce congestion observed in the neighborhood. On the

other hand, if the error is PL and change-in-error is NM, the consequent is NS to expedite the convergence of EQL to EQL_s. Thus, the packet injection period is decreased by a small amount to efficiently utilize the available wireless communication capacity.

Since an error or change-in-error may belong to up to two membership functions (MFs) in Figure 4, maximum four if-then rules may apply to the fuzzified $e(k)$ and $\Delta e(k)$. From the selected rule(s), the defuzzification interface in Figure 3 derives a single crisp control signal to adapt the packet scheduling period, if necessary, to avoid congestion as discussed next.

D. Defuzzification and Rate Adaptation

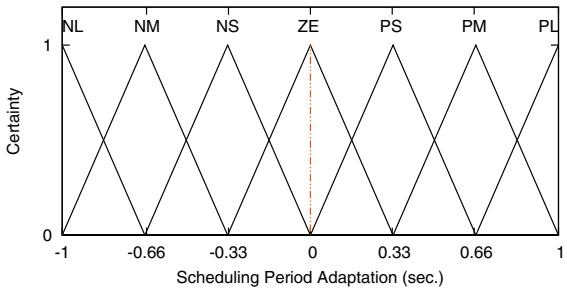

Fig. 5. Output Membership Functions for Scheduling Period Adjustment

To derive the crisp control signal, we use the MFs shown in Figure 5. Given a linguistic control signal, the corresponding crisp control signal is looked up in Figure 5. For example, if a linguistic control signal is PS in a node, the crisp control signal is 0.33 to dictate the node to increase the packet scheduling period by 0.33s.

Let i and j ($1 \leq i, j \leq 7$) represent the row and column indexes in Table I. Let $\mu(i,j)$ denote the certainty of the corresponding $rule(i,j)$ in Table I. Given the certainty values of the error and change-in-error, the certainty of the rule is equal to the *minimum* of the two certainty values. We take this approach, since the certainty of the consequent of a rule cannot be higher than its precedents. In the example given in Section V-B, the error belongs to NS in Figure 4 with certainty 1, while the *change-in-error* belongs to ZE with certainty 0.75 and PS with certainty 0.25, respectively. Hence, the rule(3,4) and rule(3,5) in Table I are selected. These rules are associated with $min(1, 0.75) = 0.75$ and $min(1, 0.25) = 0.25$ certainty values, respectively.

To convert the consequents of the selected if-then rules, we apply the *center of gravity method* [4]. Let $c(i,j)$ denote the center of the MF of the $rule(i,j)$'s consequent shown in Figure 5. For a triangular MF, the center is the middle of the triangle's base. Thus, the fuzzy control output for packet scheduling rate adaptation is:

$$\Delta p(k) = \frac{\sum_{i,j} c(i,j) \cdot \mu(i,j)}{\sum_{i,j} \mu(i,j)} \tag{8}$$

The consequents of the rule(3,4) and rule(3,5) in Table I are PS and ZE, respectively. In Figure 5, the center of PS and center of ZE are 0.33 and 0, respectively. Thus, $\Delta p(k) = (0.33 \cdot 0.75 + 0 \cdot 0.25)/(0.75 + 0.25) = 0.2475s$. Thus, the packet scheduling period is extended by $0.2475s$.

After defuzzification, the new period for scheduler invocation is computed using $\Delta p(k)$. If we let $\hat{P}(k+1) = P(k) + \Delta p(k)$, the new scheduler invocation period $P(k+1)$ is derived as follows to adapt the packet scheduling period and data generation rate at the application layer, if necessary, to avoid congestion:

$$P(k+1) = \begin{cases} \hat{P}_i(k+1) & \text{if } P_{min} \leq \hat{P}(k+1) \leq P_{max} \\ P_{min} & \text{if } \hat{P}(k+1) < P_{min} \\ P_{max} & \text{if } \hat{P}(k+1) > P_{max} \end{cases} \tag{9}$$

In this way, we bound $P(k+1)$ between P_{min} and P_{max} to support the acceptable quality of service specific to an application of interest. To consider multimedia sensing that has relatively high rates than other WSN applications relying on purely scalar sensor readings, e.g., temperature or pressure readings, we use $P_{min} = 0.001s$ and $P_{max} = 1s$ in Section VI without loss of generality.

Overall, CONSEQ only requires relatively simple event-driven computations, two sets of membership functions, and one small rule-base without consuming large amounts of computational cycles and memory space. Further, it relies on no multi-hop or e2e feedback to improve the speed of reactions to potential congestion or load imbalance, while enhancing the scalability.

VI. Performance Evaluation

In this section, we compare the performance of CONSEQ to PCCP [9] that is an efficient hop-by-hop congestion control protocol designed for multimedia WSNs. We select PCCP as the baseline because it is closest to CONSEQ in terms of hop-by-hop congestion control and event-driven congestion detection as discussed before. In PCCP, if the sink wants to receive more detailed information from a certain set of sensors, the corresponding sensors will receive higher priority. In this paper, we assume that every source has the same priority for the clarity of presentation. Extending CONSEQ to support prioritized load balancing and congestion control is reserved for future work. Especially, we have implemented CONSEQ and PCCP by extending the MiXiM [22] wireless/mobile network model provided by the OMNeT++ network simulator [8].

A. Simulation Settings

A summary of simulation parameters is given in Table II. In this paper, the radio parameters are modeled after the CC1100 ultra low power RF transceiver to show the applicability of CONSEQ to multimedia support even in a low-end WSN. The performance is expected to improve as the low-cost radio and other hardware components advance for wireless multimedia sensing [2]. As shown in Table II, we use the path loss model with the loss coefficient of 2.5. Further, the CSMA/CA protocol is used for medium access control. The MAC/network

Name	Value
Radio Bandwidth	250Kbps
Radio Carrier Frequency	315MHz
Radio Switching Time	0 (ignored)
Radio RX Power Consumption	46.5mW
Radio TX Power Consumption	86.4mW
Channel Model	Path Loss Model
Path Loss Coefficient	2.5
MAC Protocol	CSMA/CA
Max. Queue Size	25
EQL_s	12.5
Experiment Duration	60s
Number of Nodes	100+1
Data generation/source	100Kbps
Number of Source Nodes	$1-10$
Deployment Strategy	Uniform Random
Deployment Area	1000m x 1000m

TABLE II
SYSTEM PARAMETERS

layer queue can hold up to 25 packets. CONSEQ aims to ensure that the EQL in each node does not largely exceed the set-point of 12.5 packets on average.

For performance evaluation, we use uniform random grid topologies deployed over a 1000m$\times$1000m area that is divided into 25 subareas. In each subarea of 200m$\times$200m size, 4 nodes are randomly deployed and one of them can be a camera. In total, there are 100 sensor/relay nodes plus one base station located at the top-left corner. Each simulated image source generates 10 images per second to support 10 fps, similar to [1]. In our experiments, each image fits into 20 packets and each packet is 64 bytes long. Thus, each source transmits approximately 100 Kbps towards the sink in the presence of no transmission errors and corresponding retransmissions. Relay nodes (i.e., non-source nodes) forward the packets originated from the sources towards the sink. We assume that progressive image streaming, e.g. [23], is implemented at the application layer. Thus, the quality of the received image increases in proportion to the number of packets received for that image. For performance analysis, we increase the number of sources from 1 to 10. For each number of sources, we execute 10 simulation runs with different topologies and camera (i.e., source) locations. We report the average of 10 simulation runs with 90% confidence interval bars in our performance graphs.

B. Experimental Results

Our metrics for performance comparison are the (1) packet delivery ratio, (2) end-to-end delay, and (3) energy consumption. The delivery ratio is the ratio of the total number of data packets received by the sink to the total number of packets generated at the source(s). By measuring the delivery ratio, we aim to examine whether or not congestion control via rate adaptation and load balancing help improve the reliability of the e2e packet delivery. The e2e delay is the average time for a packet to travel from the application layer at the source(s) to the application layer at the sink node. It shows the performance in terms of timeliness, which affects the perceived quality of service in multimedia WSNs. Further, the energy consumption may increase as the number of retransmissions and packet losses increases.

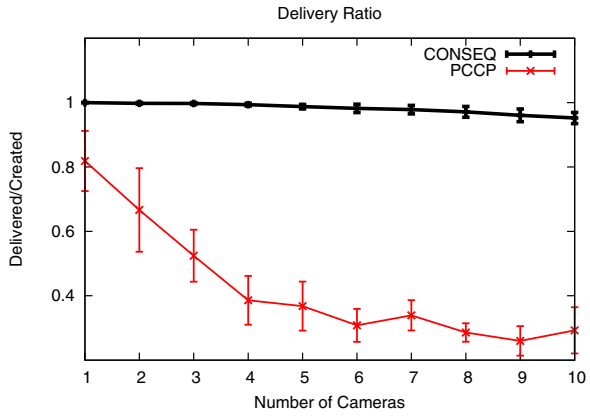

Fig. 6. Delivery Ratio

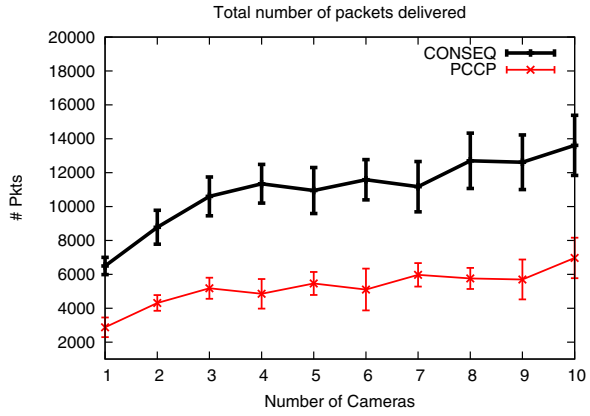

Fig. 7. Total Number of Packets Delivered

Figure 6 shows the delivery ratio as the number of sources increases up to 10. Ideally, a congestion control protocol is desired to generate/forward packet no more than the network can handle. CONSEQ achieves almost constant delivery ratio close to 100%, while PCCP drops more packets as the number of sources increases. In addition, Figure 7 shows the total number of packets delivered to the sink by CONSEQ and PCCP. CONSEQ significantly outperforms PCCP in terms of the total number of successfully delivered packets and throughput.

In Figure 8, we compare PCCP and CONSEQ in terms of e2e delay. As shown in Figure 8, CONSEQ reduces the e2e delay by up to four times. Further, it shows much smaller confidence intervals than PCCP does. Thus, its e2e delay is less variable.

Figure 9 shows the total energy consumption. Although CONSEQ's e2e delivery ratio is substantially higher than the

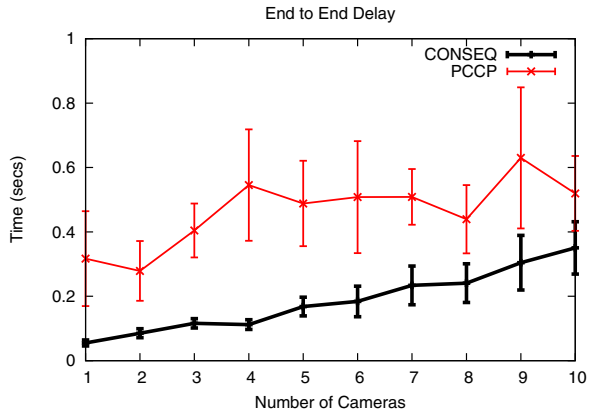

Fig. 8. End to End Delay

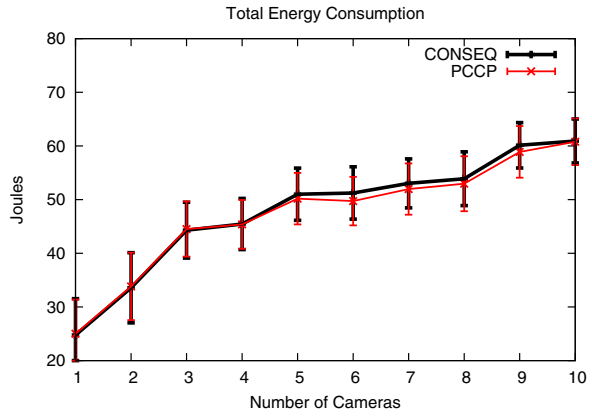

Fig. 9. Energy Consumption

delivery ratio of PCCP as shown in Figure 6, the total energy consumption of CONSEQ in Figure 9 is similar to that of PCCP. Also, compared to PCCP, CONSEQ reduces the per packet energy consumption by half for the packets successfully delivered to the sink. Overall, CONSEQ outperforms PCCP as it detects potential congestion and reacts to detected congestion early, while avoiding to generate hot-spots via load balancing.

VII. Conclusions and Future Work

The need for wireless multimedia sensing is increasing, for example, to support visual surveillance. To efficiently deal with potential congestion and packet losses in wireless multimedia sensor networks, we present CONSEQ—a new hop-by-hop approach to congestion control. Specifically, we develop two new metrics, i.e., the virtual and effective queue length, to detect potential load imbalance and congestion in each node. Moreover, each node applies fuzzy control and probabilistic load balancing to avoid creating congestion or hot-spots. In our performance evaluation, CONSEQ significantly enhances the end-to-end packet delivery ratio and reduces the end-to-end delay, while significantly reducing the energy consumption per packet successfully delivered to the sink. In the future, we will explore more advanced techniques for congestion control in WSNs.

References

[1] M. Rahimi, R. Baer, O. Iroezi, J. C. Garcia, J. Warrior, D. Estrin, and M. Srivastava, "Cyclops: in situ image sensing and interpretation in wireless sensor networks," in *Proceedings of the 3rd international conference on Embedded networked sensor systems*, 2005.

[2] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury, "A survey on wireless multimedia sensor networks," *Computer Networks*, vol. 51, no. 4, pp. 921–960, March 2007.

[3] T. Sun, L. Chen, G. Yang, M. Y. Sanadidi, and M. Gerla, "Senprobe: Path capacity estimation in wireless sensor networks," in *The Wireless Traffic Measurements and Modeling Workshop (SenMetrics)*, 2005.

[4] K. M. Passino and S. Yurkovich, *Fuzzy Control*. Addison-Wesley, 1998.

[5] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury, *Feedback Control of Computing Systems*. A John Wiley and Sons, Inc., Publication, 2004.

[6] C. L. Phillips and H. T. Nagle, *Digital Control System Analysis and Design (3rd edition)*. Prentice Hall, 1995.

[7] K. J. Åström and B. Wittenmark, *Adaptive Control*, 2nd ed. Addison-Wesley, 1994.

[8] "OMNeT++," http://www.omnetpp.org/.

[9] C. Wang, B. Li, K. Sohraby, and M. M. Daneshmand, "Upstream congestion control in wireless sensor networks through cross-layer optimization," *IEEE Journal on Selected Areas in Communications*, vol. 25, pp. 786–795, May 2007.

[10] P. Kulkarni, D. Ganesan, P. Shenoy, and Q. Lu, "Senseye: a multi-tier camera sensor network," in *ACM Conference on Multimedia*, 2005.

[11] M. H. Yaghmaee and D. Adjeroh, "Priority-based rate control for service differentiation and congestion control in wireless multimedia sensor networks," *Computer Networks*, vol. 53, no. 11, pp. 1798 – 1811, 2009.

[12] A. Woo and D. E. Culler, "A transmission control scheme for media access in sensor networks," in *ACM Conference on Mobile Computing and Networking*, 2001.

[13] D. M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN Systems*, vol. 17, no. 1, pp. 1–14, 1989.

[14] C. Y. Wan, S. B. Eisenman, and A. T. Campbell, "CODA: Congestion detection and avoidance in sensor networks," in *ACM Conference on Embedded Networked Sensor Systems*, 2003.

[15] S. Rangwala, R. Gummadi, R. Govindan, and K. Psounis, "Interference-aware fair rate control in wireless sensor networks," in *ACM SIGCOMM Symposium on Network Architectures and Protocols*, 2006.

[16] J. Paek and R. Govindan, "RCRT: Rate-controlled reliable transport for wireless sensor networks," in *ACM Conference on Embedded Networked Sensor Systems*, 2007.

[17] Y. Sankarasubramaniam, O. B. Akan, and I. F. Akyildiz, "ESRT: event-to-sink reliable transport in wireless sensor networks," in *International Symposium on Mobile Ad Hoc Networking & Computing*, 2003.

[18] C. T. Ee and R. Bajcsy, "Congestion control and fairness for many-to-one routing in sensor networks," in *ACM Conference on Embedded Networked Sensor Systems*, 2004.

[19] F. B. Hussain, Y. Cebi, and G. A. Shah, "A multievent congestion control protocol for wireless sensor networks," *EURASIP Journal on Wireless Communications and Networking*, Jan. 2008.

[20] A. Sridharan and B. Krishnamachari, "Explicit and precise rate control for wireless sensor networks," in *ACM Conference on Embedded Networked Sensor Systems*, 2009.

[21] D. Stiliadis and A. Varma, "Latency-rate servers: a general model for analysis of traffic scheduling algorithms," *IEEE/ACM Transactions on Networking*, vol. 6, no. 5, p. 611624, 1998.

[22] A. Köpke, M. Swigulski, K. Wessel, D. Willkomm, P. T. K. Haneveld, T. E. V. Parker, O. W. Visser, H. S. Lichte, and S. Valentin, "Simulating wireless and mobile networks in OMNeT++ the MiXiM vision," in *International conference on simulation tools and techniques for communications, networks and systems*, 2008.

[23] T. Wark, P. Corke, J. Karlsson, P. Sikka, and P. Valencia, "Real-time image streaming over a low-bandwidth wireless camera network," in *3rd International Conference on Intelligent Sensors, Sensor Networks and Information*, 2007.