Congestion Mitigation using In-network Sensor Data Summarization

Abu Raihan M. Kamal
School of Computer Science and Informatics
University College Dublin, Ireland
raihan@ucdconnect.ie

Chris J. Bleakley
School of Computer Science and Informatics
University College Dublin, Ireland
chris.bleakley@ucd.ie

Simon Dobson
School of Computer Science
University of St Andrews, UK
simon.dobson@st-andrews.ac.uk

ABSTRACT

Congestion can occur in Wireless Sensor Networks due to simultaneous event detection at multiple nodes, link failure or node failure. Most previously proposed congestion mitigation algorithms rely on rate control protocols to reduce network traffic. Typically, these solutions reduce application-level precision since the rate control mechanisms reduce the packet generation rate or force local packet drop without considering the implications of data loss. Herein, we propose a distributed, in-network algorithm for congestion mitigation by exploiting the inherent temporal correlation in sensor data. The proposed algorithm was implemented in TinyOS and deployed in a real-world testbed. Experimental results show that the algorithm provides significant reductions in packet drop ratio, from 25.30% to 1.92% and from 25.65% to 15.43% for temperature and light data, respectively, while incurring low distortion in the sensor data. A comparative study and network simulation were performed to assess its performance.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

General Terms

Algorithms, Design, Performance

Keywords

Sensor Network, congestion, network failure, sensor data, temporal correlation, linear regression

1. INTRODUCTION

Wireless Sensor Networks (WSNs) have become an attractive technology for diverse applications, such as environment monitoring, smart building automation, event or target detection, and health monitoring [17, 5, 20] where deployed sensors are typically constrained by computation, storage and energy. A recent trend in WSNs has been for heterogeneous networks that support source nodes with a wide range of sensor types and sampling rates introducing multiple applications on a single network [18, 3]. This trend has both increased the likelihood of congestion and increased the complexity of dealing with it.

Congestion can arise in multihop WSNs due to three factors: failures, packet storms and funneling. Experience from real-world environmental monitoring deployments, such as [9, 2, 4], indicates that *node and link failures* are common. These failures can lead to high communication loads at nearby nodes. Some WSN deployments seek to detect physical events with high accuracy. This can lead to a large number of packets being sent simultaneously from multiple nodes, i.e. a *packet storm*. Typically a WSN contains hundreds of source nodes and only a small number of Sink nodes, as a consequence there is a many-to-one network traffic flow. This *funneling* effect increases the chances of congestion in nodes close to the Sink. As well as increasing latency in the network, congestion has a negative impact on the lifetime of the network. As nodes become congested, the number of dropped packets and transmission re-tries increases significantly. This wastes energy and reduces the bandwidth of the network.

Congestion in WSNs is generally addressed in one of two ways: rate control or packet drop. *Rate control* protocols, such as RCRT [14], reduce the rate at which packets are generated at the source. *Packet drop policies* allow congested nodes to selectively drop packets [6]. Both solutions come at the cost of significant loss in application precision since the decision making nodes have no information regarding the application-level importance of data that is being discarded. Recently, Tareq at el. [1] proposed the concept of adaptive summarization whereby the congested nodes perform a predefined aggregation function to reduce the amount of data and so mitigate congestion.

Herein, we present a congestion control algorithm which uses feedback from the congested node to the source for rate control and application-level data summarization for rate reduction. Congestion detection is performed at the relay nodes and, if congestion is detected, a notification is sent back to the data source. The source reduces its packet transmission rate by means of data summarization exploiting the temporal correlations inherent in common sensor data types.

PE-WASUN'12, October 24–25, 2012, Paphos, Cyprus.

The rate is dynamically adjusted based on the status of the congested node, while minimum application-level precision is maintained. The method is lightweight in terms of computation and storage and is suitable for implementation on resource-constrained WSN nodes.

We outline our contributions as follows:

- The proposed algorithm is dynamic since it uses feedback from the congested node. Furthermore, the summarization is performed by means of linear regression which is lightweight and does not require communication between source nodes.
- A testbed network running a temperature and light intensity data collection application was deployed to evaluate the performance of the algorithm. A numerical analysis of the effect of summarization on various datasets was conducted. A comparative study and network simulation were carried out to assess the performance of the proposed algorithm.

The remainder of the paper is organized as follows. Section 2 and 3 outline related work and problem statement respectively. Section 4 describes the proposed algorithm. Evaluation of the algorithm is described in section 5. Finally, section 6 concludes the paper.

2. RELATED WORK

Typically, rate control protocols for WSN congestion can be classified as either centralized or distributed.

In the centralized case, the Sink detects congestion and uses a protocol to control it. Event-to-Sink Reliable Transport Protocol (ESRT) [15] is a centralized scheme whereby the Sink issues a congestion notification message to all sources to reduce their data rate. Similarly, in PORT [25] the Sink instructs all nodes to reduce their data sending rates based on the received packet rate. In [21], the authors provide a rate control mechanism in the downstream direction (i.e. from Source to Sink).

A distributed protocol is presented in [6] which introduces the concept of fairness using per-child packet queues. Congestion Detection and Avoidance (CODA) [23] controls the data packet generation rate using on a combination of window-based and rate-based schemes. CODA considers both an energy tax and an application fidelity penalty in its local packet drop policy. The paper defines an application fidelity penalty which only considers the data packet rate at the Sink. In [8], an algorithm based on a combination of a MAC priority queue, hop-by-hop flow control and source rate control rate is presented. These distributed solutions suffer from one major drawback: they blindly drop data packets, significantly reducing performance at the application-level.

In our previous work in [11] linear regression was exploited to prolong the lifetime of the network. A similar technique is used in this proposal for the purposes of congestion mitigation taking application fidelity into account.

To the best of our knowledge, only one previous paper considers the application precision when mitigating congestion in WSNs. In [1], the authors present a solution based on data summarization at the congested node using an error-bounded averaging technique. The work presented herein is distinct in that data summarization is performed at the source nodes which originate the data. This approach has three benefits. Firstly, summarization at the source saves energy and prolongs network lifetime. Secondly, due to intermittent links and multi-path routing, the congested node may not have all of the data from a source. Therefore, summarization may be inaccurate or impossible. Thirdly, summarizing at the source avoids placing an additional burden, in terms of receiving and processing packets, on an already congested node. This burden is compounded by the fact that the congested node may have to summarize multiple sources simultaneously.

3. PROBLEM STATEMENT

The goal of the algorithm is to detect and resolve network congestion in WSNs.

We assume that:

- The network is prone to random, sporadic congestion due to, for example, link or node failure [22].
- When congestion occurs, it persists for some time due to the existence of an underlying cause.
- Over short time intervals periodically sensed data is strongly temporally correlated at a given node, i.e., sample values can be modeled as random fluctuations on a linear trend. Temporal correlation of sensor data has been observed in previous WSN deployments such as [10, 16].

Congestion in WSN can broadly be classified into the following three types [6]:

- Type I: Since the transmission media is shared by several nearby sensors, a node may be congested due to traffic from other nodes. The MAC layer plays a vital role in mitigating such congestion. It is often achieved by ensuring fairness in shared medium access control, e.g. in [24].
- Type II: Nodes within a certain area might attempt to forward packets at the same time due to lack of local synchronization. Consequently, all nodes in the affected area become congested.
- Type III: Congestion occurs when the local buffer of a node overflows due to sudden traffic redirection to a particular node. Traffic can be redirected for many reasons, such as sudden event detection, link failure and node crash. It is important to note that the factors contributing to Type III congestion are common in various WSN deployments [22].

The scope of this work is limited to Type III congestion since Types I and II are typically dealt with by the communication protocol stack. An example of Type III congestion is provided in Figure 1. Initially the network is uncontested. Link failure (between node 4 and 7) and node failure (node 2) lead to Type III congestion at node 3. Herein, we propose that the application layer can play a vital role in mitigating Type III congestion.

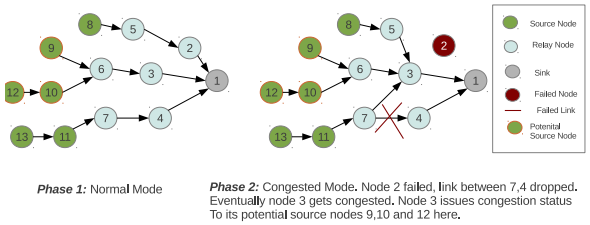

Figure 1: Congestion example

4. PROPOSED ALGORITHM

4.1 Overview

Initially, all source nodes operate in Normal Mode (N Mode), sensing and sending data according to application requirements. All transmitted packets are labelled with the unique node IDs of the source and the destination nodes. All relay nodes maintain statistics on the number of packets received per unit time from each upstream source node (see Section 4.2). All relay nodes monitor their transmit buffer-occupancy to allow congestion detection (see Section 4.3). Whenever a node detects that it is congested, it sends a control message `reduce_rate_req` to its highest rate upstream source nodes, requesting that they reduce their packet sending rate (see Section 4.4). Upon receiving the request, the source nodes switch from N Mode to Summary Mode (S Mode) (see Section 4.5). In S Mode, a source senses data in the normal way but stores the data locally, computes a summary and only sends the summary to the destination node (see Section 4.6). Meanwhile, the congested node continues to monitor buffer occupancy. If congestion persists, the congested node sends another `reduce_rate_req` message to the highest rate upstream source nodes. On receipt of this second request, the relevant source nodes further reduce their data rate. These congestion checking and rate control steps repeat until congestion is resolved. Whenever the node finds that it is no longer congested, it allows the upstream source nodes to return to N Mode (`restore_rate_req`). On receipt of this request, the affected source nodes switch back to N Mode.

The overall congestion detection and mitigation process is illustrated in Figure 2. The following sub-sections describe the algorithm in more detail.

4.2 Packet Statistics Monitoring

A lightweight method is employed for monitoring packet statistics within the network. These statistics allow congested nodes to target high rate upstream nodes in the congestion mitigation step.

All nodes maintain a Packet Statistics Table (PST) and a Packet Timer (PT). A PST entry consists of:

- `source_ID`: The packet's source node ID which is unique in the entire network.
- `count_current`: The number of packets received from that source since the most recent PT update.
- `count`: The number of packets received from the source in the last PT period, i.e. the time between the most recent PT updates.

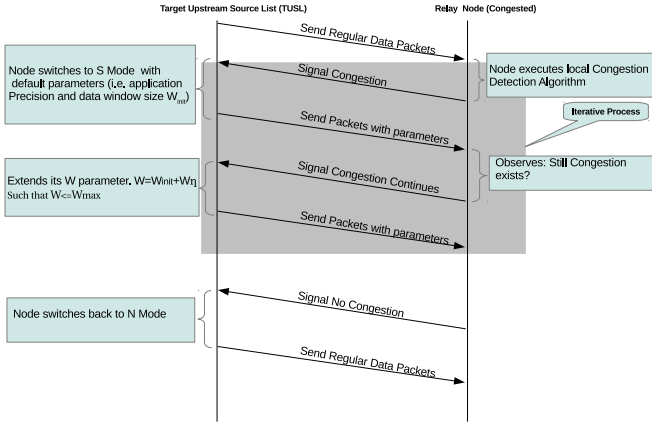

Figure 2: Operation of congestion detection and mitigation control algorithm

The PST is updated for every packet received by the node. If the source ID has not been observed before, the node creates a new record in the PST with `count_current` equal to 1 and `count` equal to zero. If the source ID has been observed, the `count` for the source is incremented. The Packet Timer (PT) updates every T_{count} seconds. On a PT update, `count` is set equal to `count_current` and `count_current` is set to zero. The number of packets received per second from each source can be calculated as $\frac{count}{T_{count}}$. The latency of these estimates is T_{count} seconds. If greater temporal resolution is needed then multiple overlapping counters N_{count} can be run in parallel, each offset by $\frac{T_{count}}{N_{count}}$ seconds.

4.3 Congestion Detection

To detect congestion, the algorithm uses a simple buffer occupancy technique. It is important to note that the proposed algorithm is independent of the congestion detection scheme. Hence alternative methods, such as [8, 23], are equally applicable.

When a packet arrives at a node, it updates its buffer status B_{status}, i.e., the *percentage of the buffer in use*. If B_{status} exceeds the threshold B_{limit} and the condition persists for T_{buffer} seconds then the node is deemed to be congested. If the traffic load of the network is short-time constant and the underlying cause of the congestion is not resolved then it is expected that congestion will persist for some time.

4.4 Congestion Mitigation

When congestion is detected, the congested node seeks to reduce the data rate of the source nodes that are sending most packets to it. The PST is searched for the IDs of source nodes with `count` values greater than N_{packet}. The sources whose `count` exceeds the threshold are sorted according to their packet counts to give the *Upstream Source List* (USL). This step eliminates low rate source nodes from the rate control procedure. Low rate source nodes include those deployed only for event detection, sources whose packets normally use other relay nodes and sources of control packets. In this way, congestion control has no adverse impact on event detection or network control. The USL building process is documented in Algorithm 1.

The top N_{target} nodes from the USL are selected as Target Upstream Source List (TUSL) for rate control. It is important to note that rate control is not governed by the congested node, rather by the linearity of the sensed data at the source node, along with the pre-defined application precision. The value of N_{target} is set *a priori* and depends on the size of the network. A `reduce_rate_req` control message is issued to each node in the TUSL. This causes the source nodes to switch from N Mode back to S Mode. The congestion node waits T_{wait} seconds. If congestion persists, the congested node repeats the process by re-sending the control message to the TUSL. This causes the source nodes to further reduce their data rates. These wait and congestion check steps are repeated until congestion is eliminated. If, after repeated rate control requests, congestion is not eliminated, the TUSL is extended by adding more nodes from the USL. The congestion mitigation procedure is repeated for these nodes. Once congestion is eliminated, the congested node reactivates the nodes in the TUSL by sending `restore_packet_rate` control messages to them. This causes the source nodes to switch from S Mode back to N Mode. The restore messages are sent one-at-a-time starting from the bottom of the list with a pause of T_{wait} seconds between each. This slow restore ensures that the network does not become unstable, switching between congested and uncongested states. If congestion re-appears, the slow restore process is terminated and the congestion mitigation process re-starts, maintaining knowledge of the source nodes still under rate control.

Algorithm 1 Upstream Source List

```
1: Variables: MinExpected; Expected no. of minimum packets for each node
   USL : the list to be built, initially empty.
2: while currentTime ≤ SW do
3:    // On reception of each data packet p_i:
      n_i.totalpacket = update()
4: end while
5:
6: if n_i.totalpacket ≥ MinExpected then
7:    USL.addNode(n_i)
8: end if
   USL.sortList() ////sort the nodes (w.r.t. no. of packets)
9: for i = 1 to USL.Length() do
10:   n_i = getNode(USL) & set n_i.rank = i
11: end for
```

4.5 Rate Control

On initialization, source nodes operate in N Mode according to the following default data collection parameters for the application:

- `sensor_ID`: Sensor ID. A single node may support multiple sensors.
- `type`: Sensor data type, e.g. temperature, humidity.
- T_s: Sensing period.

In addition, source modes are configured with the default settings of the following rate control parameters:

- δ: Application precision. This parameter bounds the error arising from data summarization.
- W_{init}: Initial window size. This parameter is the number of samples included in a single summary. A higher value of W_{init} means a lower packet rate but, typically, a greater summarization error.
- W_{max}: Maximum window size. This parameter limits the number of samples in a data summary.
- W_η: Window size increment.

The starting values of all parameters are set *a priori* by the end-user or application developer. On initialization, the current window size W is set to W_{init}.

When a source node receives a `reduce_rate_req` control message, it switches to S Mode. When a source node receives subsequent `reduce_rate_req` control messages, it increases W by W_η, subject to the limit that $W \leq W_{max}$. When a source node receives a `restore_rate_req` control message, it switches back to N Mode and sets W to W_{init}.

A source node that is not sending data periodically may receive a `reduce_rate_req`. In this case, a `reduce_rate_reject` is sent to the congested node. The congested node then eliminates the source ID from its TUSL and adds the next node from the TUSL.

4.6 Data Summary

For a given node and sensor, the sensed data can be represented as a matrix D:

$$D = \begin{bmatrix} t_0 & d_0 \\ t_1 & d_1 \\ t_2 & d_2 \\ ... & .... \\ t_{W-1} & d_{W-1} \end{bmatrix}$$

where t_i is the sampling time for row i, d_i is the data sampled at time t_i and W is the current window length.

Clearly, the sampling times can be summarized using the time of the first sample t_0, the sampling period T_s and the window length W. Here, we propose to summarize the sampled data d_i using the following equation:

$$\hat{d}_i = \beta_1 i + \beta_2 \tag{1}$$

where the coefficients β_1 and β_2 are determined by Linear Regression over the matrix D [13].

Herein, we use the Least Square Method (LSM) to determine the values of β_1 and β_2 that minimize the total square error between the straight line and the measured data.

$$\beta_1 = \frac{S_{id}}{S_{ii}} \tag{2}$$

$$\beta_2 = \bar{d} - \beta_1 \bar{i} \tag{3}$$

where $\bar{i}$, the mean of i, is equal to $t_0 + \frac{(W-1)T_s}{2}$, $\bar{d}$ is the mean of d_i, and

$$\begin{aligned} S_{id} &= \sum_{i=0}^{W-1} (i - \bar{i})(d_i - \bar{d}) \\ &= \sum_{i=0}^{W-1} (i - \frac{W-1}{2})(d_i - \bar{d}) \end{aligned} \tag{4}$$

$$S_{ii} = \sum_{i=0}^{W-1} (i - \bar{i})^2 \quad (5)$$

Once β_1 and β_2 have been calculated, the quality of the fit is assessed using the *Pearson product moment correlation coefficient* r defined as:

$$r = \frac{S_{id}}{\sqrt{S_{ii} S_{dd}}} \quad (6)$$

where

$$S_{dd} = \sum_{i=0}^{W-1} (d_i - \bar{d})^2 \quad (7)$$

The value of r is always between -1 to 1. The greater the absolute value of r, the stronger the linearity in the data and the better the fit. The value obtained is compared to the application-specific precision parameter δ. If $|r| \geq \delta$ then the summarization is deemed successful (i.e. $LinearTest = 1$) and the data summary, listed in Table 1, is sent to the destination. If summarization fails ($LinearTest = 0$) the node switches back to N Mode. The node tries to switch back to S Mode in the next time window subject to the condition that congestion persists.

On receipt of the summary data, the destination node expands the summary data according to Equation 1 to obtain an estimate of the original sensed data.

The overall data summarization algorithm is described in Algorithm 2.

Table 1: Summary data.

Field	Meaning
`source_ID`	Source node ID
`sensor_ID`	Sensor ID
`type`	Sensor data type
t_0	Time-stamp of the first sensed data
t_s	Sensing period
W	Window size in samples
β_1	Linear regression coefficient 1
β_2	Linear regression coefficient 2

Algorithm 2 : Linear Test

1: Input: D, δ
2: Output: status=[1,0]
3: Initialize Global Variables: $\bar{x} = 0, \bar{y} = 0$ status=0;
4: $S_{id} = ExtractValue(D)$ // equation 4
5: $S_{ii} = ExtractValue(D)$ //equation 5
6: $S_{dd} = ExtractValue(D)$ //equation
7: $r = \frac{S_{id}}{\sqrt{S_{ii} \times S_{dd}}}$
8: **if** $|r| \geq \delta$ **then**
9: status=1
10: **else**
11: status=0
12: **end if**

5. EVALUATION

This section reports the results of an assessment of the performance of the proposed algorithm both in a testbed implementation and in a numeric simulation on various sensor datasets. A network simulation and a comparative study were also carried out.

5.1 Implementation

The proposed algorithm was implemented in TinyOS using nesC as the programming language [19]. TinyOS version 2.1.1 and nesC compiler version 1.3.3 were used. Crossbow TelosB nodes were used in the testbed. The Collection Tree Protocol [7] was used for routing.

5.2 Implementation Results

Two programs were written and installed separately on motes. *Program 1* was for a source node with regular data collection functionality, while *Program 2* was for a source node supporting both N Mode and S Mode. Table 2 gives the memory usage results. The ROM size of *Program 2* was 12% greater than that of *Program 1*, mainly due to the inclusion of the `math.h` library for statistical computation. The results show that the software has little impact on the source nodes in terms of memory footprint.

Table 2: Program Size

	ROM Size (in Bytes)	RAM Size (in Bytes)
Program 1	30,346	1,914
Program 2	33,998	1,918
Increase (%)	12.03%	.021%

5.3 Testbed Performance

To assess the performance of the proposed algorithm, a small testbed network was deployed, as shown in Figure 3. In the deployment, the source nodes sense the ambient temperature every 150 ms and send the sensed values to the Sink. The high sampling rate was specifically chosen to cause congestion. The testbed experiences Type III congestion due to the high rate of data collection and funneling. The duration of the experiment was 1 hour per iteration. Different parameter settings where used in each iteration. In each iteration, the Packet Drop Rate (PDR) was measured for each node in the network every 10 minutes. At the same time, the mean PDR was calculated over all nodes in the network. PDR is defined as the ratio of the total number of packets received by Sink to the total number of packets generated by all nodes within a specified time period.

In the first iteration, all source nodes ran in N Mode. In the second iteration, congestion control was enabled and the value of N_{target} set to 2, i.e. 2 nodes are switched to S Mode when congestion is detected. Subsequently, two further iterations were conducted with $N_{target} = 3$ and $N_{target} = 4$. In all cases, the congestion control parameters were set as listed in Table 3.

The experiment was repeated for light intensity sensing during the day. The source nodes were placed outside the window of the laboratory.

The PDR obtained is given in Figure 4 and the mean values provided in Table 4. Clearly, the proposed algorithm was effective in reducing the PDR, particularly with larger val-

Table 3: Testbed Congestion Control Parameters

Parameters	Value
`type`	Temperature, Light
T_s	150 ms
δ	0.9
W_{init}	30
W_{max}	100
W_η	10
N_{target}	varied from 2 to 4

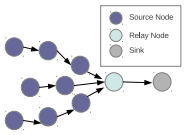

Figure 3: Testbed: Network Layout

ues of N_{target}. The improvements in PDR for temperature and light data with $N_{target} = 4$ were 23.38% and 10.22%, respectively. The algorithm was more effective in the case of temperature data (PDR dropping to 1.92%) than for light intensity (PDR above 15%) since the sensed temperature data was more strongly temporally correlated than the light intensity (day-time). Hence, linear fitting was more accurate.

Table 4: Testbed Results

Data:Temperature	
Operation Mode	Mean PDR (%)
S Mode Only	25.3
with S Mode, N_{target}=2	16.15
with S Mode, N_{target}=3	7.34
with S Mode, N_{target}=4	1.92
Data:Light (Day-time)	
Operation Mode	Mean PDR (%)
S Mode Only	25.65
with S Mode, N_{target}=2	23.22
with S Mode, N_{target}=3	20.5
with S Mode, N_{target}=4	15.43

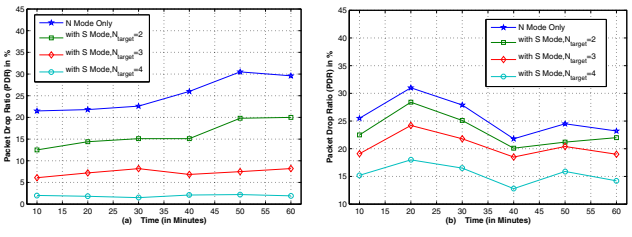

Figure 4: PDR in Testbed: (a): Temperature Data (b): Light Data

5.4 Data Summarization Performance

Since the correlation coefficient r is directly related to the degree of congestion mitigation or PDR, this numerical analysis was performed on various data sets to show the effects of other parameters such as `type` and W on r. Matlab was used for computational purpose. Outdoor sensor data for humidity and light were collected for two days with sensing intervals of 15 seconds. On average, a total of 11,500 data points were accumulated. The entire data series was subdivided into smaller *samples* where each data sample's length was bounded by the value W. In other words, each *sample* contained consecutive W data points.

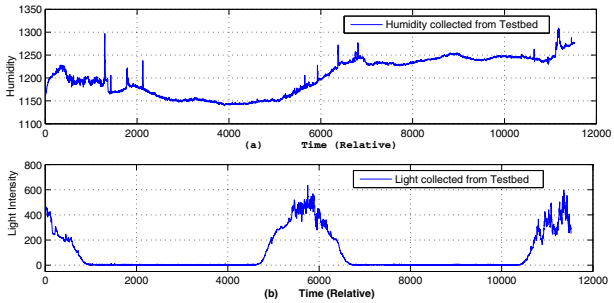

Figure 5: Linearity of Sensor data from the testbed, a) Humidity b) Light Intensity (both day and night)

For the experiment 100 *samples* were chosen randomly from each data set with the specific value of W. Figure 5 shows that light for night-time showed the best linearity since it was almost equal (dark) for the entire night. And it was reversed for day-time light data. Humidity showed a moderate degree of linearity. We also varied the value of W, a higher value ensures reduced traffic but also provides less probability of a high correlation coefficient r. The histogram in Figure 6 shows the values of r. In Figure 6 (a) with W=30 we obtained the lowest range of r between 82.5% and 87.5% for 17% samples while in Figure 6 (b) with W=50 it was between 67.5% to 72.5% for 12% samples. Figure 6 (c) and (d) show the extreme effects of day and night on light data. In Figure 6 (c) we noticed that 100% of the samples r values above 87.5% whereas in Figure 6 (d) more that 90% samples had r values below 80%.

In Figure 7 results from the four data sets are presented using Empirical Cumulative Distribution Function (ECDF). In this figure the graph away from y-axis is better in terms of linearity. Light data shows two extremes in the figure. These results indicate linearity in sensor network data depends on several issues such as type of data, place of deployment and time of data collection.

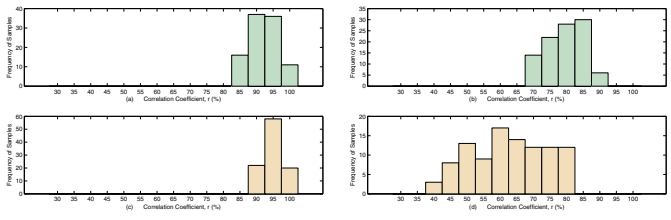

Figure 6: Histogram of Pearson Correlation Coefficient r for : (a)Temperature data with W=30 (b)Temperature data with W=50 (c) Night-time Light intensity with W=30 (d) Day-time Light intensity with W=30

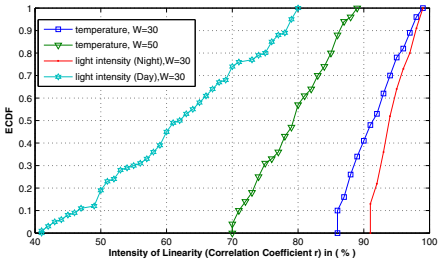

Figure 7: Pearson Correlation Coefficient r on various data sets

5.5 Comparative Study and Network Scalability

Comparison. To the best of the authors' knowledge, only one other paper considers application fidelity and congestion control in a combined way. In [1] data summarization was performed at the congested node using estimation. In our proposal, data summarization was performed at the source node using a Linear Regression Method. We argue that this proposed scheme incurs less energy compared to *Congestion Control for Spatio-Temporal Data* since in our proposed scheme the source computes the parameters and only forwards the parameters. We use the following formula [12] for calculating total energy E:

$$E = B \times C \sum_{i=1}^{L} t_i$$

where, B=size of the data packet in bytes, C=unit cost for transmitting or receiving, L=no. of hops to traverse to reach the node. We set B=10 bytes, C=1 and L varies from 2 to 10. Window size W was set to 30. We varied the *linear probability* p_i of $LinearTest = 1$ from .9 to .6. Numerical analysis of the sensor data in the previous section reveals high variation in the degree of correlation. Hence we use the term *linear probability,* p_i to make the comparison more realistic.

The finding is shown in Figure 8(a) which indicates a linear increase of energy E with the source distance. Comparing with the previous scheme, in our proposed framework energy is significantly reduced (by a factor of 7) for high value of $p_i = .9$. Even with $p_i = .6$ it is reduced by a factor of 2 compared with the previous scheme.

Scalability. To assess the scalability of the proposal, a network simulation was performed using TOSSIM [19].

Since TOSSIM uses dummy sensor data, the $LinearTest$ function was modified to omit the $|r| \geq \delta$ condition, i.e., data is always summarized in S Mode. A simulation script was written in Python 2.7. The simulation parameters are presented in Table 5. Table 6 contains congestion parameters. Initially the network topology was uniform, then an iterative process was performed to remove nodes and links so that the overall PDR drops due to Type III congestion. Finally simulation was run with *S Mode* options in specific nodes. The number of nodes in S mode is proportional to the size of network. The selection of nodes to switch to *S Mode* was performed randomly.

Table 5: Simulation Parameters

Parameters	Value
PHY Layer	2.4GHz IEEE 802.15.4
MAC Layer	Default CSMA
Routing Layer	CTP
External Noise	Yes (Mayer Library)
Total No. of Nodes	10 to 100
Total No. of Nodes in S Mode	25% of Network Size
No. of BS/Sink	1 (ID=1)
Sim. Duration	10^{10} sim events
No. of Iteration	5

Table 6: Congestion Parameters

Parameters	Value
W_{init}	30
W_{max}	100
W_{η}	10

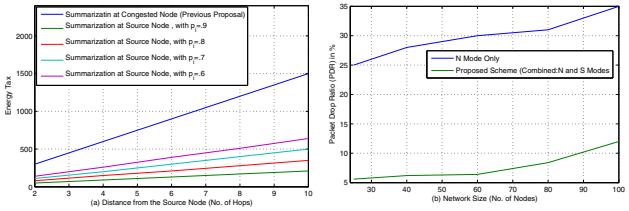

Figure 8: (a) Comparive Study (b) PDR in Simulation

Figure 8(b) shows the effectiveness of the algorithm in reducing PDR. It also reveals that the proposal ensures a linear increase in PDR with respect to network size.

6. CONCLUSION

This paper presents a novel and distributed algorithm for congestion control in WSNs which exploits the inherent linearity of sensor data to reduce the data rate by means of summarization. The data rate reduction is bounded by the application-level data precision requirements.

The algorithm was implemented in a real-world testbed network and in simulation. In a testbed experiment PDR was found to drop from 25.3% to 1.92% for high rate temperature data, and for light data it dropped from 25.65% to 15.43%. A network simulation in TOSSIM was carried out to show the scalability of the solution. Simulation results indicate that the improvements in PDR are roughly constant with network size. In the future, we plan to investigate more sophisticated techniques for selecting sensor nodes considering both data and network status to apply rate control protocol.

Acknowledgment

This work is partially supported by Science Foundation Ireland under grant number 04/RPI/1544, "Secure and Predictable Pervasive Computing" and Higher Education Authority PRTLI4 under grant number R10891, "NEMBES: Networked Embedded Systems."

7. REFERENCES

[1] H. Ahmadi, T. F. Abdelzaher, and I. Gupta. Congestion control for spatio-temporal data in cyber-physical systems. In *Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems*, ICCPS '10, pages 89–98, New York, NY, USA, 2010. ACM.

[2] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A line in the sand: A wireless sensor network for target detection, classification, and tracking. *Computer Networks (Elsevier*, 46:605–634, 2004.

[3] S. Bhattacharya, A. Saifullah, C. Lu, and G.-C. Roman. Multi-application deployment in shared sensor networks based on quality of monitoring. In *Proceedings of the 2010 16th IEEE Real-Time and Embedded Technology and Applications Symposium*, RTAS '10, pages 259–268, Washington, DC, USA, 2010. IEEE Computer Society.

[4] P. Buonadonna, D. Gay, J. M. Hellerstein, W. Hong, and S. Madden. Task: Sensor network in a box. In *In Proceedings of European Workshop on Sensor Networks*, pages 133–144, 2005.

[5] O. Chipara, C. Lu, T. C. Bailey, and G.-C. Roman. Reliable clinical monitoring using wireless sensor networks: experiences in a step-down hospital unit. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pages 155–168, New York, NY, USA, 2010. ACM.

[6] C. T. Ee and R. Bajcsy. Congestion control and fairness for many-to-one routing in sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pages 148–161, New York, NY, USA, 2004. ACM.

[7] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, pages 1–14, New York, NY, USA, 2009. ACM.

[8] B. Hull, K. Jamieson, and H. Balakrishnan. Mitigating congestion in wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pages 134–147, New York, NY, USA, 2004. ACM.

[9] F. Ingelrest, G. Barrenetxea, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange. Sensorscope: Application-specific sensor network for environmental monitoring. *ACM Trans. Sen. Netw.*, 6:17:1–17:32, March 2010.

[10] Intel lab sensor data, 2004. http://db.csail.mit.edu/labdata/labdata.html.

[11] A. R. M. Kamal, M. A. A. Razzaque, and P. Nixon. 2pda: two-phase data approximation in wireless sensor network. In *Proceedings of the 7th ACM workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, PE-WASUN '10, pages 1–8, New York, NY, USA, 2010. ACM.

[12] H. Luo, H. Tao, H. Ma, and S. K. Das. Data fusion with desired reliability in wireless sensor networks. *IEEE Trans. Parallel Distrib. Syst.*, 22(3):501–513, Mar. 2011.

[13] W. Mendenhall and T. Sincich. *Statistics for Engineering and the Science*, chapter 11, pages 531–646. Prentice-Hall, NY, 4th edition, 1994.

[14] J. Paek and R. Govindan. Rcrt: rate-controlled reliable transport for wireless sensor networks. In *Proceedings of the 5th international conference on Embedded networked sensor systems*, SenSys '07, pages 305–319, New York, NY, USA, 2007. ACM.

[15] Y. Sankarasubramaniam, O. B. Akan, and I. F. Akyildiz. ESRT: event-to-sink reliable transport in wireless sensor networks. In *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 177–188, New York, NY, USA, 2003. ACM.

[16] EPFL SensorScope Project. `http://sensorscope.epfl.ch/index.php/Environmental_Data`, 2008. [Online accessed: Nov-10-2010].

[17] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler. An analysis of a large scale habitat monitoring application. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, pages 214–226, New York, NY, USA, 2004. ACM.

[18] A. Tavakoli, A. Kansal, and S. Nath. On-line sensing task optimization for shared sensors. In *IPSN '10: Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 47–57, New York, NY, USA, 2010. ACM.

[19] TinyOS Documentation. `http://docs.tinyos.net/index.php/Main_Page`, 2010. [Online accessed: Jan-10-2010].

[20] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong. A macroscope in the redwoods. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, SenSys '05, pages 51–63, New York, NY, USA, 2005. ACM.

[21] R. Vedantham, R. Sivakumar, and S.-J. Park. Sink-to-sensors congestion control. *Ad Hoc Netw.*, 5(4):462–485, May 2007.

[22] M. Wachs, J. I. Choi, J. W. Lee, K. Srinivasan, Z. Chen, M. Jain, and P. Levis. Visibility: a new metric for protocol design. In *Proceedings of the 5th international conference on Embedded networked sensor systems*, SenSys '07, pages 73–86, New York, NY, USA, 2007. ACM.

[23] C.-Y. Wan, S. B. Eisenman, and A. T. Campbell. Energy-efficient congestion detection and avoidance in sensor networks. *ACM Trans. Sen. Netw.*, 7(4):32:1–32:31, Feb. 2011.

[24] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks, 2002.

[25] Y. Zhou, M. R. Lyu, and J. Liu. Port: A price-oriented reliable transport protocol for wireless sensor networks. In *In International Symposium on Software Reliability Engineering (ISSRE*, pages 117–126, 2005.