Overview

• Congestion in Wireless Sensor Networks
• Effects of Congestion
• Congestion Control
• Challenges of Congestion Control
• Examples of Congestion Control in WSNs
  • IFRC
  • RCRT
Congestion in Wireless Sensor Networks

- In WSNs many nodes typically gather data to a single destination (or a small number of nodes)
- Applications that transport large volumes of data can cause congestion on nodes
- CSMA-based MACs
- Do TDMA-based MACs have such problems as well?
Applications that Require Congestion Control

• High data rate applications

• Event driven applications with high sampling frequency (Structural monitoring)
Effects of Congestion

- Packets can be dropped at congested node’s internal queue
- Fairness between different source nodes can break
Congestion Control

• Techniques that control the traffic flow on nodes to avoid/mitigate/resolve congestion
Challenges of Congestion Control

- Detecting and notifying interfering nodes
- Fairness control
- Avoid fluctuation
- Maximize throughput
- Find optimal rate with minimal delay
- Low overhead preferred
Classification

- Distributed
- IFRC
- Centralized
- RCRT
Classification

- Adaptive rate allocation
- Many congestion control techniques are based on AIMD
- Directly compute target rate
IFRC

• Interference-Aware Fair Rate Control in Wireless Sensor Networks

• S.Rangwala et al.

• SenSys 2006
Main Question!

What distributed mechanisms must a WSN have to allocate ‘fair’ and ‘efficient’ rate to each node in the network
IFRC

• Assumptions
  • CSMA/CA MAC layer
  • MAC layer level retransmissions
  • Tree routing protocol exists
  • The topology is stable
IFRC

• Fair

  • Each flow receives ‘at least’ the most congested fair share rate

• Efficient

  • Less congested parts in the tree are allowed to have higher rates
To allocate rates, define ‘potential interferers’

A node N1 is a potential interferer of node N2 if a flow originating from N1 uses a link that interferes with the link between N2 and its parent

Larger set than neighboring nodes
IFRC

Consider a network of sensor nodes, with each node uniquely identified by an integer in the range 1, ..., N. Specifically, we assume a sink tree rooted at node 10, as shown in Figure 1. The sink tree is a spanning tree connecting the base station, node 10, to all other nodes in the network. Each node in the tree represents a sensor node, and the links between nodes represent wireless communications.

In the simplest version of IFRC, node 10 is the root node, and it sources a flow labeled 'r'. This flow traverses the sink tree to reach the base station. The flow is retransmitted at node 10 if it is not acknowledged by the base station. The retransmission priorities are used to determine the order in which flows are transmitted over the channel.

Each node in the sink tree independently sources flows to deliver data to the base station. The flows are multiplexed over the available channel capacity, and each node's flow is assigned a priority based on its retransmission history. The flow with the highest priority is transmitted first, and the channel capacity is divided among the flows in proportion to their priorities.

We assume that the MAC layer provides link-layer retransmissions recover from most packet losses. Outside this regime, IFRC needs an end-to-end feedback mechanism to determine when retransmissions are necessary. To leverage some of this work in the design of an end-to-end reliability mechanism that would complement IFRC.

Finally, we assume that the MAC layer provides link-layer retransmissions. This data can traverse multiple hops before reaching the base station. The sink tree is stable, and changes in the underlying routing tree.

In Figure 1, solid lines represent the sink tree, and dashed lines represent potential interferers. The sink tree is a set of nodes where each node is a child of its parent node. Potential interferers are nodes that are neither a parent nor a child of a node. The sink tree is designed to adapt to changes in the network topology, such as node failures or link failures.

By contrast, this scheme's non-work-conserving scheduler does not promote efficiency; nodes whose flows do not pass through the congested regions cannot fully utilize the available bandwidth. IFRC forces overall fairness by using a non-work-conserving weighted-fair queue for each of its children. Besides mechanistic differences, IFRC extends easily to token-based and TDMA MACs.
IFRC

• Rate allocation
  • Start at base rate
  • Use AIMD scheme to increase rate when channel conditions are good and decrease rate when congestion happens
IFRC

- Components
  - Measuring congestion levels
  - Congestion sharing
  - Rate adaptation
IFRC

• Measuring congestion levels
  • Measure occupancy of queues (EWMA)
  • Set upper and lower thresholds to set congestion state and cancel out congestion state
  • Rate is halved when detecting congestion
IFRC

• Congestion sharing

• Each outgoing packet contains...
  • current rate
  • average queue length
  • congestion bit
  • smallest rate among congested children (node i)
  • Node i’s average queue length
IFRC

- When congestion information is received, set the rate to the lower of my current rate and the congested neighbor’s rate
- NEVER exceed parent’s rate
- Base station will send explicit packets to notify one hop nodes of congestion in the tree
IFRC

- Rate adaptation
  - Similar to TCP’s slow start
  - AIMD-based increase every time interval and halve when congested
• Evaluation
  • TinyOS 1.1
• 40 Mote testbed
• Modify multihopLQI protocol for routing
we have discussed in Section 5.

shows the number of packets received at the sink as a function of experiments, we also present a per-flow illustration of IFRC on a 40-node network. All 40 nodes are transmitting data, showing poor parameter choices for IFRC.

Figure 5: Routing tree and link qualities used in the experiment. We plot the change in rate, at each network node. In addition, we logged every packet received at the base station. This fairly detailed log-ling helps us visualize IFRC behavior in several ways. However, this is beyond our control, of course, but we compensate during and across runs, the quality of wireless links can and does change. Preliminary experiments, we have found IFRC to work well in dynamic topologies. However, in order to study the steady-state behavior of IFRC, we modified the existing TinyOS routing protocol design, demonstrate that IFRC can be extended in the ways presented in Section 5.

In Table 1, we show the parameters used in our experiments (unless otherwise shown) reveals that the rate adaptation at different nodes is slightly greater than the diameter of the underlying topology. This surpris-

Figure 6: Per flow goodput in the 40-node experiment

Table 1: Parameters used in our experiments (unless otherwise shown).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Additive Increase Parameter</td>
<td>$\text{MAX}$</td>
</tr>
<tr>
<td>Slow-start mult. incr.</td>
<td>$0.025$ per pkt</td>
</tr>
<tr>
<td>Lower Threshold</td>
<td>$0.0025$ pkt/sec</td>
</tr>
<tr>
<td>Upper Threshold</td>
<td>$0.025$ pkt/sec</td>
</tr>
<tr>
<td>EWMA Weight</td>
<td>$0.79$</td>
</tr>
<tr>
<td>Max Packet Size</td>
<td>$32$ bytes</td>
</tr>
<tr>
<td>Min Packet Size</td>
<td>$4$ packets</td>
</tr>
<tr>
<td>Upper Threshold</td>
<td>$8$ packets</td>
</tr>
<tr>
<td>Lower Threshold</td>
<td>$64$ pkts</td>
</tr>
<tr>
<td>Mobility Rate</td>
<td>$0.94$</td>
</tr>
<tr>
<td>Node Id</td>
<td>$0.79$</td>
</tr>
<tr>
<td>Node Id</td>
<td>$0.87$</td>
</tr>
<tr>
<td>Node Id</td>
<td>$0.89$</td>
</tr>
<tr>
<td>Node Id</td>
<td>$0.91$</td>
</tr>
<tr>
<td>Node Id</td>
<td>$0.93$</td>
</tr>
<tr>
<td>Node Id</td>
<td>$0.95$</td>
</tr>
<tr>
<td>Node Id</td>
<td>$0.96$</td>
</tr>
</tbody>
</table>

Figure 6: Per flow goodput in the 40-node experiment.
This results from IFRC's aggressive rate cutting at each conducted, we have never seen a single instance of buffer overflow due to queue overflow. In fact, buffer size of 64, and, so, in this experiment, no packets are lost a fair bit, but never grows beyond 20. Each of our nodes has a sustainable on the routing tree of Figure 9.

While it is difficult to decipher the detailed behavior of each flow beyond our control.

sults from an infrequent data logging queue overflow problem at IFRC, but kept buffer sizes the same as in our above experiment, in all the experiments that we have conducted, we have never seen a single instance of buffer overflow due to queue overflow. In fact, buffer size of 64, and, so, in this experiment, no packets are lost a fair bit, but never grows beyond 20. Each of our nodes has a sustainable on the routing tree of Figure 9.

Figure 9: Instantaneous queue size in 40-node experiment

Figure 10: Optimality test for the 40-node experiment

Figure 8: Rate adaptation in the 40-node experiment

Packet Count

Time (Hour:Min)

Figure 7: Packet reception in the 40-node experiment
IFRC

Figure 8: Rate adaptation in the 40-node experiment

Figure 9: Instantaneous queue size in 40-node experiment
IFRC

Figure 13: Per flow goodput with weighted fairness

Figure 14: Per flow goodput with only a subset of senders

Figure 15: Per flow goodput with multiple sinks

Figure 16: Per flow goodput with no link-layer retransmissions
RCRT

- RCRT: Rate Controlled Reliable Transport in Wireless Sensor Networks
- J.Paek et al.
- Sensys 2007
RCRT

• Centralized approach to resolve congestion given that many WSNs gather data to a single base station and the base station has global knowledge of entire network

• End-to-end reliability

• Tolerates dynamic topologies
• Base station determines the rate for each node and sends back feedback packets
• Base station determines rate based on a given ‘policy’
RCRT

- Feedback packets
  - When packets are properly received no action is taken
  - When packets are lost (out of order) base station asks for missing packets
  - Keep estimated RTT of packet retrieval
  - Measure RTT of retransmissions
  - If measured RTT is significantly higher, congestion exists in the path
  - Contains allocated rate for each node and missing packets
RCRT

- Base station sends AIMD based rate assignments
- Periodically increase rate
- Reduce rate when congestion is detected
RCRT

• Base station sends AIMD based rate assignments
  • Periodically increase rate
  • Reduce rate when congestion is detected
RCRT

- Policies
  - Demand proportional
  - Demand limited
  - Fair
- Used to determine rate of individual nodes
RCRT

• Evaluation
RCRT

Figure 3—Rate $r_i$ allocated to every node in the 40-node experiment with fair rate policy

Figure 4—Per-flow goodput in the 40-node experiment with fair rate policy

Figure 5—Packet reception plot for all nodes in the 40-node experiment with fair rate policy

Figure 6—Percentage of packet repaired by end-to-end loss recovery mechanism in the 40-node experiment
Consider a single-source network. We conducted a single-source experiment. The baseline experiment demonstrates some of the salient features of RCRT’s algorithms. The next question we address is: how close does RCRT's rate allocation get to the ideal? One way to evaluate the performance achieved by RCRT is: how close does RCRT's rate allocation get to the ideal? Another way to evaluate RCRT's optimality is to compare its achieved rate with that of Figure 8. If we define 0.9 pkt/sec as the maximum sustainable rate for reliability, but no congestion control. Our network is able to sustain up to 0.9 pkt/sec per node. Thereafter, it experiences congestion collapse: acknowledgments and retransmission use up much of the network capacity, resulting in less goodput. If we compare the achieved rate with that of Figure 8, we can quantify the overhead of feedback in RCRT. Of course, since RCRT is congestion-adaptive, sources only send at the assigned rates, not at their demands. The figure plots this increasing demand on the x-axis.

The small spikes below the curve represent that all nodes have approximately fair goodput throughout the experiment. The error bars parallel to the y-axis indicate the maximum and minimum goodput among all nodes.

**Figure 7**—Average goodput achieved by best-effort transport. X-axis is the rate at which each node sourced packets. Y-Error bar represent the maximum and minimum goodput among all nodes.

**Figure 8**—Average goodput achieved by reliable transport. X-axis is the rate at which each node sourced packets. Y-Error bar reprint the maximum and minimum goodput among all nodes.

**Figure 9**—Average goodput achieved by RCRT. X-axis is the demand assigned to each node. Y-Error bar represent the maximum and minimum goodput among all nodes.
RCRT

Figure 10—Rate $r_i$ allocated to each node in the 40-node experiment with demand-proportional rate allocation policy when 8 nodes join in after 500 seconds

Figure 11—Per-flow goodput in the 40-node experiment with demand-proportional rate allocation policy when 8 nodes join in after 500 seconds
other hand, RCRT fully utilizes the network queues until packets, IFRC detects incipient congestion and aggressively detects congestion that IFRC aggressively. A second reason is functionality at the sink, which has a more global view of advantage comes from implementing its congestion control. The first, of course, is that much of RCRT's performance the rate achieved by IFRC: 0.293 pkts/sec. 0.824 pkts/sec in this experiment, which is more than twice experiment.

Figure 15 shows the rate achieved by IFRC together with -Rate achieved by IFRC and RCRT in 30-node experiment

Figure 15—Rate achieved by IFRC and RCRT in 30-node experiment