Routing Without Routes
The Backpressure Collection Protocol

Scott Moeller, Avinash Sridharan, Bhaskar Krishnamachari,
Omprakash Gnawali
Viterbi School of Engineering
University of Southern California

October 15, 2010
Wireless Sensor Networks

- Static Networks of Low-Power Wireless Embedded Devices
- Wide range of Applications: Building Automation, Industrial Process Control, Automated Meter Reading, Environment Monitoring, Precision Agriculture, Asset Tracking and Localization
Motivation

Challenges

- Because of high density deployments, external interference, and limited bandwidth, network protocols must provide good source rates.
- Even in static network deployments, there are significant link dynamics due to fading and external interference, sink mobility.
- Routing and rate control protocols for Wireless Sensor Networks (WSN) must be kept low complexity.
Motivation

Challenges

- Because of high density deployments, external interference, and limited bandwidth, network protocols must provide good source rates.
- Even in static network deployments, there are significant link dynamics due to fading and external interference, sink mobility.
- Routing and rate control protocols for Wireless Sensor Networks (WSN) must be kept low complexity.

This work in a nutshell:

- We present the Backpressure Collection Protocol, a novel routing approach that does hop-by-hop per-packet forwarding rather than computing end-to-end paths. It is simple to implement and shows excellent rate performance and robustness to dynamics.
- We also present an implementation of a simple alpha-fair source rate controller that adds congestion control capability to BCP.
Routing and forwarding based on distributed weight computations:

\[
 w_{i,j}(t) = \left( [Q_i(t) - Q_j(t)] - V \cdot ETX_{i,j}(t) \right) \cdot R_{i,j}(t)
\]

**Routing Control Decision:**
Node \(i\) identifies the outbound link with greatest weight \(w_{i,j^*}\).

**Forwarding Control Decision:**
If \(w_{i,j^*} > 0\) then forward the packet, else wait time \(T\).
How do packets find their way to the sink?

\[
    w_{i,j}(t) = (Q_i(t) - Q_j(t)) - V \cdot ETX_{i,j}(t) \cdot R_{i,j}(t)
\]

The sink is the only node that can pull the packets from the network

- The sink has zero queue backlog
- Link weight computations generate gradients toward the sink
- Per-hop queue differentials are impacted by link cost

A simple linear example

\[
    V \cdot ETX_{i,j} = 1 \quad R_{i,j} = 1
\]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>S</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>S</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>
\[ w_{i,j}(t) = (Q_i(t) - Q_j(t) - V \cdot ETX_{i,j}(t)) \cdot R_{i,j}(t) \]
Handling Dynamics

\[ w_{i,j}(t) = \left( [Q_i(t) - Q_j(t)] - V \cdot ETX_{i,j}(t) \right) \cdot R_{i,j}(t) \]
Handling Dynamics

\[ w_{i,j}(t) = \left( [Q_i(t) - Q_j(t)] - V \cdot ETX_{i,j}(t) \right) \cdot R_{i,j}(t) \]
**Theory Background**

**Theoretical Relationship**
- Backpressure routing of BCP is a distributed approximation to the centralized queue backpressure scheduling

**Theoretical Origins: Lyapunov Drift-Based Stochastic Optimization**
- Based on work by Tassiulas and Ephremides ’92
- Extended by Neely ’03, Georgiadis et al. ’05, to include utility optimization

**Objective**
Minimize time average expected system transmissions while maintaining strongly stable queues:

\[
\min f(\bar{x})
\]

\[
\text{s.t. } \limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}[Q_i(\tau)] < \infty \text{ for all } i
\]
Translating from Theory to Practice

BCP is the First Ever Systems Implementation of Backpressure Routing

- We have implemented the first ever backpressure routing protocol for many-to-one wireless sensor networks.
- Written for TinyOS 2.x, a large market share embedded operating system
- Implemented over IEEE 802.15.4 compliant radios

Packet Looping

Addressed by using ETX as a link penalty

Packet Delivery Delay

Addressed by use of LIFO service priority

Scalability

Support for queue scaling through Floating Queues
Static Network Testbed Configuration

Network Parameters

- Motes 1-40 on Tutornet
- 802.15.4 channel 26
- Transmit power -18 dBm

- Sink mote 1
- Source motes 2-40
- Poisson arrivals
Comparison with the Collection Tree Protocol

We benchmark BCP against the state-of-the-art Collection Tree Protocol [Gnawali et al., Sensys 2009] (CTP) for TinyOS 2.x.

Packet Overhead

The header added by BCP is 8 bytes, the same size as that used by CTP. BCP does not use periodic beacon mechanisms, and therefore has no beacon control overhead.
Max-Min Achievable Goodput

**Figure:** Per source Goodput versus source rate.
Max-Min Achievable Goodput

Figure: Per source Goodput versus source rate.

Figure: Maximum queue size over 35 minute experiment at 0.5 and 1.5 packets per second per source.
A Comparison of Packet Delivery Efficiency

- Source-to-Sink transmissions per packet delivered
  - Using only data-driven link estimation, BCP performs competitively with CTP
  - Static network packet transmission efficiency is not a factor in improved network capacity
Delivered Packet Delay

Persistent minimum backlogs and FIFO service priority impacts delivered packet delay tremendously.
An Intuitive Motivation for Our LIFO Innovation

A

\[ V \cdot ETX_{A \rightarrow B} \]
\[ V \cdot ETX_{B \rightarrow C} \]
\[ V \cdot ETX_{C \rightarrow S} \]

Node A B C S

B

Arrival

Node A B C S

C

Arrival

Node A B C S
Closure of the Low Rate Delay Gap Using LIFO

Figure: Delivered packet delay CDF for node 4 (1 hop, top) and 40 (4 hops, bottom) at 0.25 PPS/Source. System average delivered packet delay is reduced by more than 98% through LIFO usage.
The Floating Queue: Our Support for Scalability

![Graph showing performance metrics for floating queues with and without floating.]

- **Goodput**: The top graph illustrates the goodput for floating queues compared to those without floating. The x-axis represents node IDs, and the y-axis shows the goodput values.
- **Avg Q Size**: The middle graph displays the average queue size for floating queues versus those without floating. It also includes a line for the maximum data queue size without floating.
- **Avg Tx / Pkt**: The bottom graph shows the average transmission per packet for floating queues and those without floating.

These metrics help in understanding the scalability benefits of using floating queues in network routing.
The Floating Queue: Our Support for Scalability

Floating Queue Operation
- Finite data queue
- Data Q overflows: discard to virtual Q
- Data Q underflows: service virtual Q
Experiment Setup

External interference
- 2 Devices
- 802.11 channel 14
- 20 sec on / 10 sec off
- 890 Bytes x 200 PPS Each

Sources and Timeline
- 0.25 PPS / source
- Interference on @ 300 Sec
- Interference off @ 1200 Sec
High Sink Mobility Performance

Sink Mobility

- 20 mote sink sequence
- 1,000 ms / sink
- 0.25 PPS / source
High Sink Mobility Performance

Sink Mobility

- 20 mote sink sequence
- 1,000 ms / sink
- 0.25 PPS / source
High Sink Mobility Performance

Sink Mobility

- 20 mote sink sequence
- 1,000 ms / sink
- 0.25 PPS / source

<table>
<thead>
<tr>
<th>Mobility</th>
<th>BCP</th>
<th>CTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delivery Ratio</td>
<td>0.996</td>
<td>0.590</td>
</tr>
<tr>
<td>Average Tx/Packet</td>
<td>1.73</td>
<td>9.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Static</th>
<th>BCP</th>
<th>CTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delivery Ratio</td>
<td>0.969</td>
<td>.999</td>
</tr>
<tr>
<td>Average Tx/Packet</td>
<td>2.39</td>
<td>2.65</td>
</tr>
</tbody>
</table>
A Free Lunch?

There remain some important areas in need of future investigation:
- Learning time
- Out-of-order packet delivery
- Low power operation under asynchronous sleep cycling

BCP is Available

- Source Code is in TinyOS Contrib (usc/bcp)
- [http://anrg.usc.edu/~scott/](http://anrg.usc.edu/~scott/)
The need for congestion control

Stability is not guaranteed
Burst events can cause queue sizes to overcome ETX penalties
Subsequent poor routing decisions cause network collapse
### Alpha Fair Rate Control for BCP

**Alpha Fair Rate Control Utility Function**

\[
max \sum_i (1 - \alpha)^{-1} (r_i)^{1-\alpha}
\]

**Rate Computation**

\[
\lambda_{alpha} = V_{alpha} \cdot \frac{1}{[Q(t)]^{1/\alpha}}
\]

### Strengths

- Simply implemented atop BCP (~ 200 lines of code)
- Very small code footprint (892 Bytes ROM, 384 Bytes RAM)
- No additional packet overhead
20 Mote Alpha-Fair rate control performance
40 Mote Alpha-Fair rate control performance

![Graph showing the performance of the 40 Mote Alpha-Fair rate control protocol. The graph compares the goodput (PPS), average transmission per packet (Tx/Pkt), and delivery percentage with different settings and rates. The data is presented for two rates: 1.33 pps and 1.33 pps with V_alpha=2.15.](image-url)
Parameter sensitivity to topology

![Graph showing parameter sensitivity to topology](image)

- **Goodput (PPS)**
  - $V_{\alpha}=2.5$ (Red)
  - $V_{\alpha}=5$ (Green)

- **Avg Tx/Pkt**
  - $V_{\alpha}=2.5$ (Red)
  - $V_{\alpha}=5$ (Green)

- **Delivery %**
  - Values range from 85% to 100%

The graph illustrates the impact of different parameter values on network performance metrics such as goodput, average transmission per packet, and delivery percentage.
Parameter selection for Alpha-Fair

- $\alpha = 4$ provides reasonable fairness emphasis in 20 and 40 mote empirical tests
- $V_{ETX} = 2$ performs well across all empirical tests
- $V_{alpha}$ must be tuned dynamically; this is an open problem we are currently investigating.

Alpha-Fair rate controller performance atop BCP

- Zero packet overhead (no packet header enlargement)
- For tuned $V_{alpha}$, (95%,80%) of empirical max-min rate is achieved on (20/40) motes
Conclusions

A network+transport backpressure stack for wireless sensor networks

- We presented BCP: the first implementation of dynamic backpressure routing in any network
- BCP shows excellent rate performance and robustness to dynamics
- We also presented the implementation of an alpha-fair rate control that works with this dynamic routing mechanism.

Future Directions

- Interoperability with low-power asynchronous sleep protocols for WSN
- We’re starting to work on an extended implementation for 802.11-based mesh/ad-hoc networks, including support for any-to-any routing