A PSFQ Implementation Study

Manu Bansal*, Bhaskaran Raman*
Computer Science and Engineering
Indian Institute of Technology Kanpur

Introduction

Pump Slowly Fetch Quickly (PSFQ) is a reliable transport protocol for Wireless Sensor Networks (WSNs) that appeared in 2002 with simulation results. We make a comprehensive implementation study of the protocol on the Tmote platform (with TinyOS) for throughput and scalability. In the process, a generic implementation of PSFQ has been developed.

A number of issues surfaced during implementation that were not addressed in the design, and have a significant impact on the performance of the protocol. We also study the protocol in light of the Bridge Monitoring (BrMon) WSN Application being developed at IIT Kanpur, which was the original motivation for this work, and justify a custom transport protocol Pump Slowly Fetch Slowly (PSFS) for the application.

Implementation

Network layout

Base Protocol

B-BS PSFQ communication replaced by a simple base (transport) protocol called BProto because using PSFQ required maintaining a complete PSFQ implementation on the base node (in C) communication over UART is error-free - a trivial protocol would not be degrading

Component Hierarchy

Independent modules for each layer of functionality
Application code kept separate; uses simple interfaces from the transport layers
Filesystem back-end uses Rincon’s Blackbook (TinyOS-1.x, contrib/rincon), a generic filesystem implementation

Implementation Design

Bundle: transport layer data unit, which is a collection of MAC layer data units called segments
Segmentation and reassembly handled by PSFQ

Experiment Methodology

Study Space

• PSFQ implementation vs. simulation results
• PSFQ vs. PSFS (on BrMon data profile)

Performance Metrics

Throughput/Latency
Delivery overhead
Delivery ratio

Network Parameters

Channel properties
Topology
Protocol parameters

Also of interest is the independent best performance of PSFQ for a given unicast network (linear topology) and a given packet error rate. This is exactly the analysis required for comparing PSFQ vs. PSFS.

Hence, the independent best performance analysis of PSFQ is contained in comparison with PSFS.

Experimental Setup

Experimentation under lab-settings; no attempt towards realistic distance separation or direction
• Maximum radio power levels, hence maximum possible interference, simulated channel error

Performance Analysis

<table>
<thead>
<tr>
<th>Parameter</th>
<th>PSFQ Sim</th>
<th>PSFQ Impl</th>
<th>PSFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radio</td>
<td>2 Mbps</td>
<td>256 Kbps</td>
<td>256 Kbps</td>
</tr>
<tr>
<td>MAC</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Ch. Desc</td>
<td>Linear</td>
<td>Linear</td>
<td>Linear</td>
</tr>
<tr>
<td>Topology</td>
<td>Linear</td>
<td>Linear</td>
<td>Linear</td>
</tr>
<tr>
<td>Plc Size</td>
<td>1500 pkts, 10B datagram</td>
<td>500 pkts, 2kB datagram</td>
<td>500 pkts, 2kB datagram</td>
</tr>
<tr>
<td>File Size</td>
<td>2400B, 100 pkts</td>
<td>1000B, 50 pkts</td>
<td>2400B, 100 pkts</td>
</tr>
<tr>
<td>Data Cache</td>
<td>RAM</td>
<td>RAM</td>
<td>RAM</td>
</tr>
</tbody>
</table>

Configuration comparisons

PSFQ vs. PSFS

Conclusions

Latency results similar to simulation for no error
Latency does not vary with packet error rate for 1-hop; latency rises faster than simulation for multi-hop, mainly due to interference
Transfer beyond 20-30% error could not be achieved for 3-hop and higher, implementation corner cases, difficult implementation
Transmission rate too low for overlap to become tolerable
PSFS always better than PSFQ for considered network sizes

Performance gain may result at very high error rates in absence of overlap - amounts to not utilizing the designed pipelining; gain insignificant even for 1-hop
Performance will deteriorate even more in case of simultaneous multiple transfers due to increased overlap
Implementation requires a non-trivial flash integration for caching requirements such that caching is not a bottleneck

Acknowledgements

I am deeply thankful to Dr. Bhaskaran Raman, IIT Kanpur for his enriching and thought-encouraging guidance. I acknowledge the contribution of Mr. Barsaian and Prateek Gupta, IIT Kanpur during the early stages of this work. I thank Nilesh Mishra and Hemanth Haridas for filling in with their knowledge and experience. Finally, I thank the sponsors of WISARD ‘07 and Dr. Bhaskaran Raman for making this presentation possible.

* (manub, braham}@cse.iitk.ac.in