PSFQ : A Reliable Transport Protocol for wireless Sensor Networks

by Chieh-Yin Wan,Andrew T. Campbell, Lakshman Krisnamurthy

Presented by **Smaragdakis Georgios** during BU Sensor Seminar

Motivation

- There is an emerging need to be able to re-task or reprogram **groups** (clusters) of sensors in **wireless** (high error rate) sensor networks on the fly
  (eg. During disaster recovery).
- Due to the application-specific nature of sensor networks, it is **difficult to design a single monolithic** transport system that can be optimized for every application.

The Problem

Data that flows from sinks to sources for the purpose of control or management (eg. re-tasking sensors) **is sensitive to message loss**. Loss of single message associated w/ code segment or script would render the image useless and the re-tasking operation fail.

The proposed Protocol: PSFQ

The KEY idea is to design a reliable,light protocol to:

Distribute data from a source node by pacing data at a relatively slow speed – that is **Pump Slowly**/Smoothly (PS).

&

Missing segments from immediate neighbors very aggressively using local recovery – that is **Fetch Quickly** (FQ).

Protocol Design

• Messages that are lost (mainly due to wireless link**!!**) are **detected when a higher sequence number than expected is received** at a node triggering the fetch operation.

•**Minimize the number of transmissions** for lost detection and recovery operations with minimum signaling

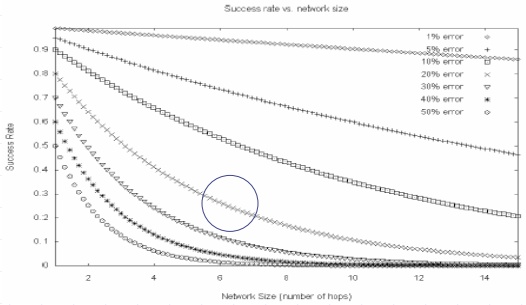
• **Operate correctly** even in an environment where the radio link quality is very poor

• **Provide loose delay bounds** for data delivery to all the intended receivers.

Error Recovery (DON'Ts)

Error accumulates exponentially over multi-hops

If packet loss of wireless channel is p then the chances of exchanging a message successfully across a single hop is (1-p).



The probability that a message is successfully received across n hops decreases quickly to $(1-p)^n$.

Error Recovery

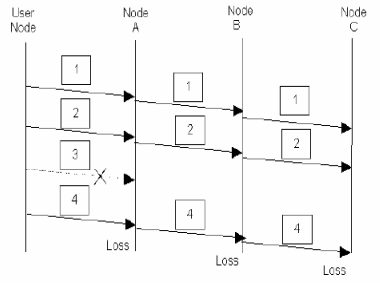
A **hop-by-hop Recovery** :

The chances of exchanging a message successfully across a single hop is **(1-p) and is scalable (independent of # of nodes)**

Multi-modal Operations

- **NOT Only Forwarding**:

  Loss event will keep on propagating until TTL=0 and packet is dropped, BUT it is fast and better for error free environments.



- **NOT Only Stop-And-Forward**

  Large Delays (even for local use) BUT the only choice in highly error-prone environments.

Multi-modal Operations

PSFQ Solution: **In sequence data forwarding**
(a.k.a localizing loss events and not relaying any higher sequence number messages until recovery has taken place).

Cost: Cashes in intermediate nodes and keep state in each node.

Benefits: The pump mechanism operates in a multi-hop packet forwarding mode during periods of low errors when lost can be recovered quickly and behaves more like store-and-forwarding communications when the channel is highly error-prone.

Technical Details – Pump Operation

Node broadcasts a packet to its neighbors every Tmin.

Receivers checks for gaps in sequence number (Data cash used for duplicate suppression).

If this is a new packet (decrease TTL => TTL==0) and there is no gap in seq# then a transmission is scheduled.

Constraint: **Tmin< Ttransmit < Tmax** ,(Tmin,Tmax timers)

Benefits: **By delaying transmission, quick fetch operations are possible AND Reduction of redundant transmissions**.

Technical Details - Fetch Operation (a.k.a Relay-initiated Error Recovery)

A node goes into Fetch mode once a seq# gap in a file fragments is detected and in that case Node will send a NACK message upstream ONLY

It is possible that more than one packet is lost before a node detect loss, so a **"window"** of lost packets is used.

Receivers **randomize** their NACK transmissions and

cancel NACK if the node "overhear" NACK for the same segment from a neighbor in order to reduce redundancy.

If there are still gaps NACKs are generated every Tr (Tr is a Fetch timer). Constraint: **Tr < Tmax**

Technical Details – (ProActive)Fetch Operation (a.k.a Limits of Fetch)

In case that the **last segments of a file** are lost the loss detection is impossible because no next segment exists.

Solution: **Node enters 'proactive fetch' mode** if last segment has not been received and no packet has been delivered since Tpro (Tpro is a new timer).

Tpro = a * (Smax-Slast) * Tmax , a>=1 and Smax-Slast is the number of remaining segments associated with this file

Tpro must not be too short (wasted control msgs) and **not too large** (increase delivery latency of the entire file).
And **Tpro=a * k * Tmax**, k fragments can be kept in cash.

Technical Details – Report Operation

(a.k.a Selective Status Reporting)

This is a **feedback/monitoring mechanism**

In order to reduce the communication cost (sending one packet instead of sending the same amount of data using small packets) Only the **last hop responds immediately**. The other (hop-by-hop) nodes piggyback their state info when they receive the report delay.If there is no space left in the msg, then a new msg will be created.

For not last hop: **Treport = Tmax*TTL + Δ** , Δ random

Revision : Timers!

Tmin< Ttransmit < Tmax

Tr < Tmax

Tpro = a * (Smax-Slast) * Tmax

Treport = Tmax*TTL + Δ

Performance Evaluation: Simulation

NS-2 simulator to compare **PSFQ with SRM** (Scalable Reliable Multicast) without using IP multicast substrate but using omniscient multicast routing scheme (SRM-I).

• Metric1 : **Average Delivery Ratio**: # of msgs a target node received, to the # of msgs a user node injects into the network.

• Metric 2 : **Average Latency**: Average time elapsed from the transmission of the first data packet from the user node until the reception of the last packet by the last target node in the sensor network.

• Metric 3 : **Average Delivery Overhead**: The total # of msgs sent per data msg received by a target node.

Parameters and Topology

Scenario : Re-tasking of a simple sensor Sensor Network in a Disaster Recovery Scenario within a Building

The user at location 0 inject a program into the SN of that floor.

•Uniformly distributed channel error model

•Radio: 2Mbps, 25 m range (100mx100m area), simple CSMA/CA

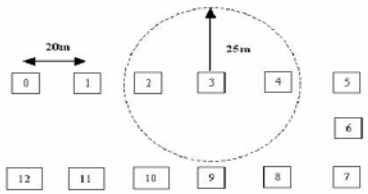
•Image file=2.5K, packet size=50 bytes (50 packets total)

•Transmission rate: 1 packet/10 ms

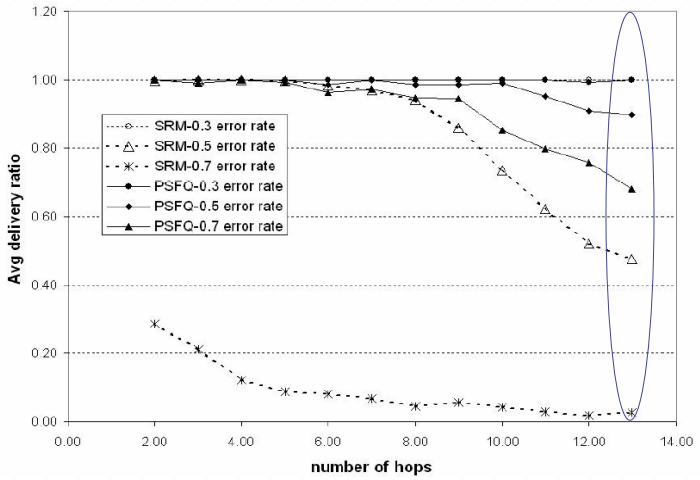
•Tmax = 100ms

• Tmin = 50 ms

•Tr = 20 ms



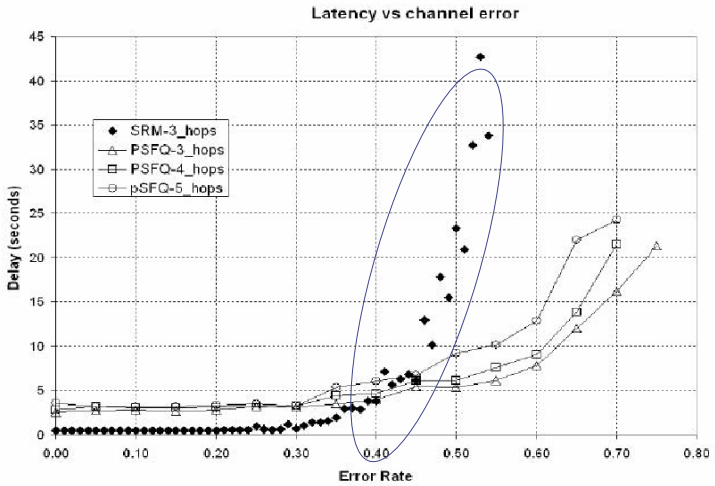
Results- Error Tolerance



SRM-I 100% delivery at : 13 hops at 30%er, 5 hops at 50%er, 0 hops at 70%er

PSFQ 100% delivery at : 13 hops at 30%er, 10 hops at 50%er, 4 hops at 70%er

Results- Latency



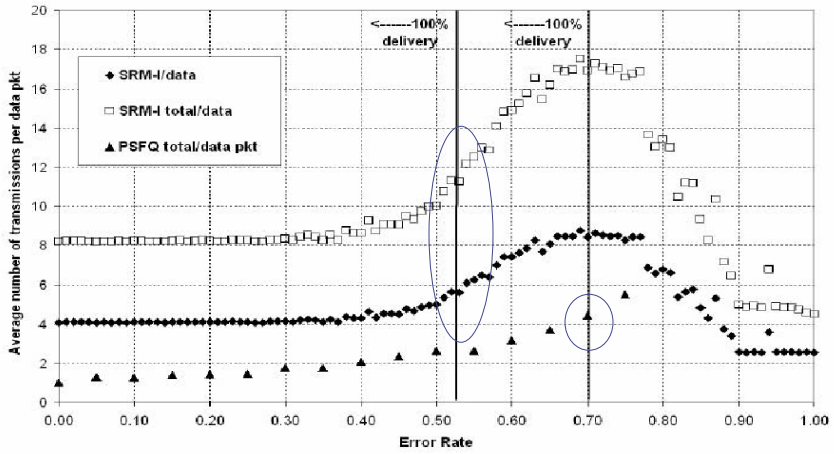
SRM-I: Decreased latency at low error rates, exponential increase after 40%

PSFQ : Higher latency at low error rates, better handling at higher rates

'Pump slowly' increases latency at low error rates

Advantage: performance degradation is graceful at higher rates

Results- Communication Cost



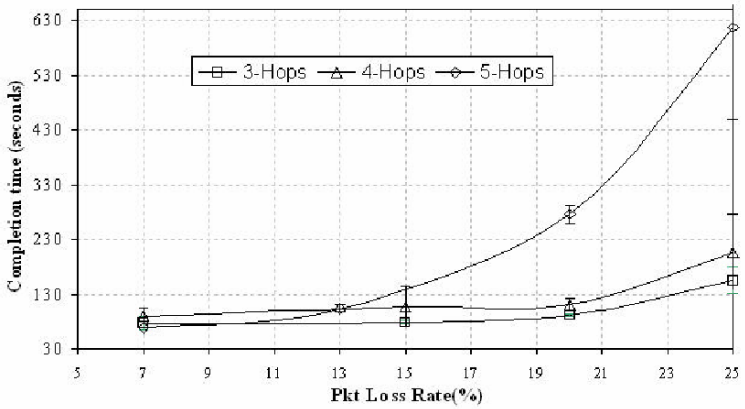
3HOP Network (measured average delivery overhead).

SRM-I: More overhead in all cases (100% delivery limit at 52%er)

–Reason: periodic exchange of session messages for loss detection/recovery (… at 70%er)

PSFQ : Less overhead due to 'on-demand' loss recovery scheme

Experimental Results



Implementation of PSFQ using the TinyOS platform on RENE motes (nodes:ATMEL 4MHz 8 bit µcontrls,8KB progMem and 128KB EEPROM).

Much poorer than simulation: exponential increase in delay happens at 11% loss rate or higher.