PSFQ : A Reliable Transport protocol for wireless Sensor Networks

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

Presenter
Vishal Kumar Singh
Columbia University

Reliability – Motivation and Need

- Application specific needs for reliable communication e.g. Biomedical sensors, Structural monitors. [Source to Sinks]

- Re-tasking or reprogramming Sensors e.g. Upgrading Software or Algorithms, Adding codes, scripts etc. [Sink to Source]

- Control Information e.g. Query, Management information e.g. message to change duty cycle.

PSFQ : Pump Slowly Fetch Quickly

- Key protocol features (What are the main contributions ?)

  - Independent of Routing Layer.
  - Power conserving (Hop by Hop error recovery).
  - Power conserving (uses NACK – no NACK implosion problems).
  - Multimodal Operation (Adaptive for low and high error rate)
  - Provide loose delay bounds for data delivery.

PSFQ : Pump Slowly Fetch Quickly

- Key protocol features
  - High error tolerance.
  - Scalable and Efficient (Aggregation of Errors, minimum retransmission and retransmission requests).
  - Low Signaling overhead.
  - Customizable protocol.
- Assumptions (Are they reasonable – Why ?)
  - No Congestion in the network.
  - Message loss caused by lossy links. What does the assumptions lead to ?

PSFQ : Pump Slowly Fetch Quickly

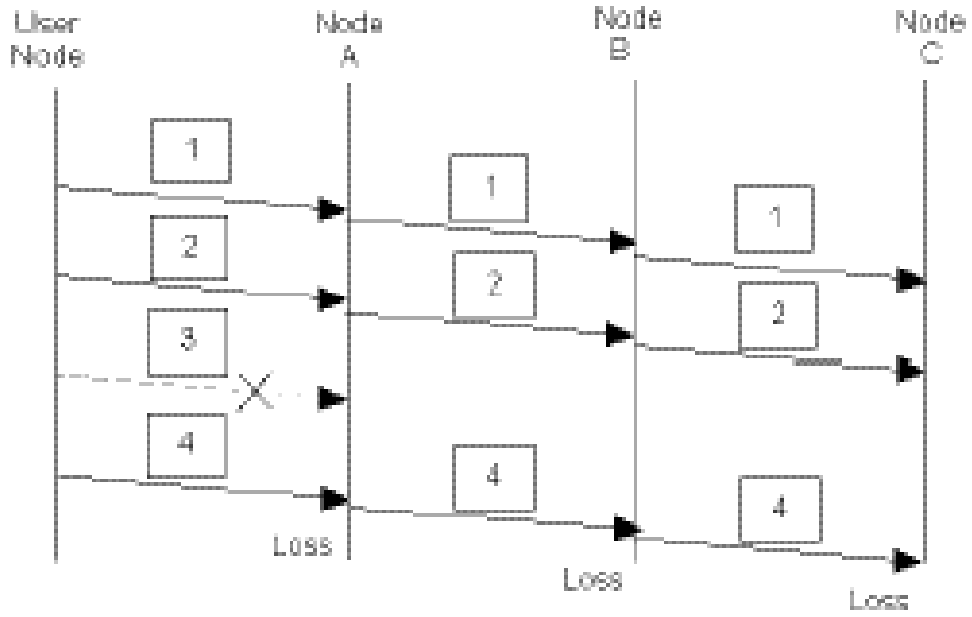
- Working
  - Distribute data slowly (pump slow)
  - Recover from errors quickly (Fetch Quick).
  - Missing data recovered from immediate preceding neighbor. [One hop error recovery, NACK based]
  - Lost messages detected by seeing a higher sequence number then expected.

PSFQ : Pump Slowly Fetch Quickly

- Working : Multimodal Operation
  - Packet-forwarding mode – At low error rates.
  - Store-and-forward mode – At high error rates.
  - Prevents Loss event propagation – by in-sequence data propagation.
  - Localize loss events.

PSFQ : Pump Slowly Fetch Quickly

- Multimodal Operation : Prevent Loss event propagation



PSFQ : Pump Slowly Fetch Quickly

- Operations : Pump
  - Pump starts with Inject Message – File Id, File length, Seq No, TTL and data (payload).
  - Receiver checks for missing sequence number.
  - Receiver uses data cache for duplicate suppression.
  - No retransmission if heard 4 times (as no coverage benefit) – Reduction in number of Transmissions.

PSFQ : Pump Slowly Fetch Quickly

- Operations : Pump
  - Source Node broadcasts every Tmin.
  - Receiver decrements TTL and schedules to fwd at any time between Tmin and Tmax – Delaying by Tmin allows for quick fetch operation.
  - Statistical Delay (Loosely Bound) Delay = Tmax * n * number of hops.

PSFQ : Pump Slowly Fetch Quickly

- Operation : Fetch
  - Fetch mode starts when a gap/missing sequence number is detected.
  - Node sends NACK message which has File Id, length and Loss window.
  - Aggregation of Loss Windows (seq num pairs).
  - NACK is not propagated beyond 1-hop.(no implosion)

PSFQ : Pump Slowly Fetch Quickly

- Operation : Fetch
  - Cancel NACK if node overhears same NACK or repair request from some other node.
  - NACKs are generated every Tr (randomized) until all gaps are recovered or retries over. Tr<Tmax (pump-fetch ratio).
  - Allows recovery of segments in loss window from the neighbors as they schedule a reply if they hear a NACK and have missing segment in there data cache between 0,Tr.

PSFQ : Pump Slowly Fetch Quickly

- Operation : Pro-Active Fetch
  - Loss of last or all packets.
  - Proactive fetch if no packet received for time Tpro. Tpro = a* (Smax - Slast) * Tmax.
  - For nodes with limited cache Tpro =Tmax * n *a .

PSFQ : Pump Slowly Fetch Quickly

- Operation : Report
  - Used for feedback
  - Only the last hop will respond immediately.
    - Piggybacking
    - New reports send before previous reports so that new nodes do not create new reports.
    - Treport =Tmax * TTL for a node.
  - What Information does the report contain ? Possibly can be used to free data cache.
  - How do u think protocol can be improved ? Any thing which you see can be more efficiently or in a better way?

PSFQ : Pump Slowly Fetch Quickly

- Simulation : results

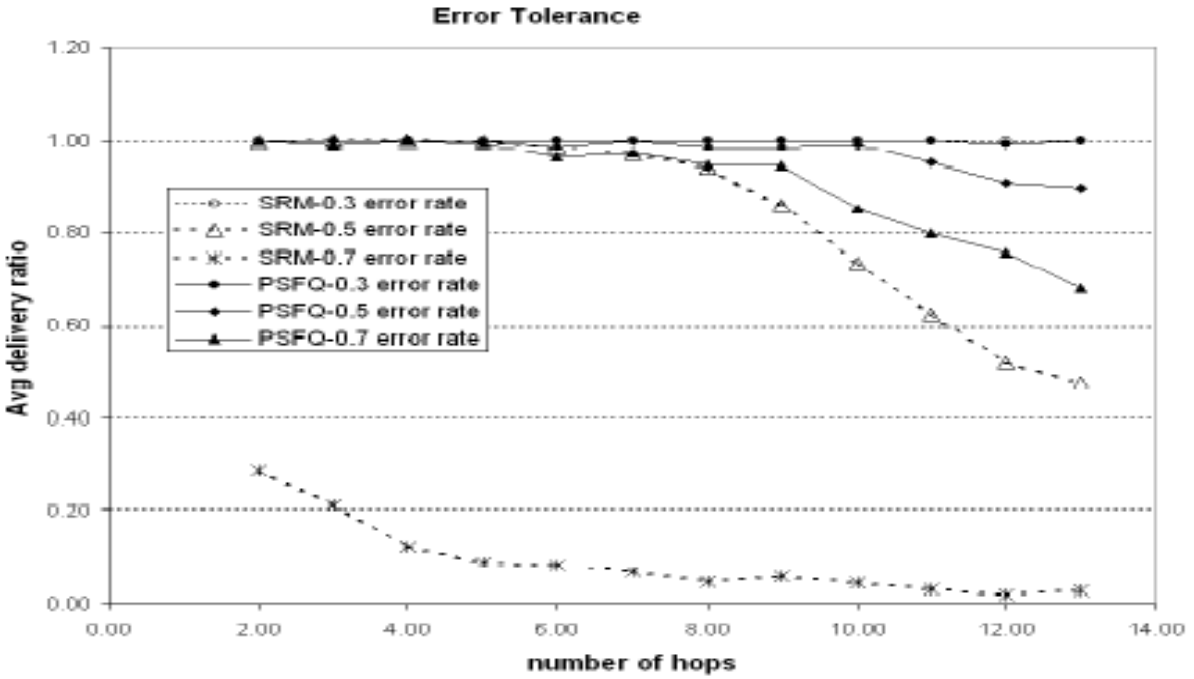


Figure 5. Error tolerance comparison - average delivery ratio as a function of the number of hops under various channel condition for different packet error rate.

PSFQ : Pump Slowly Fetch Quickly

- Simulation : results

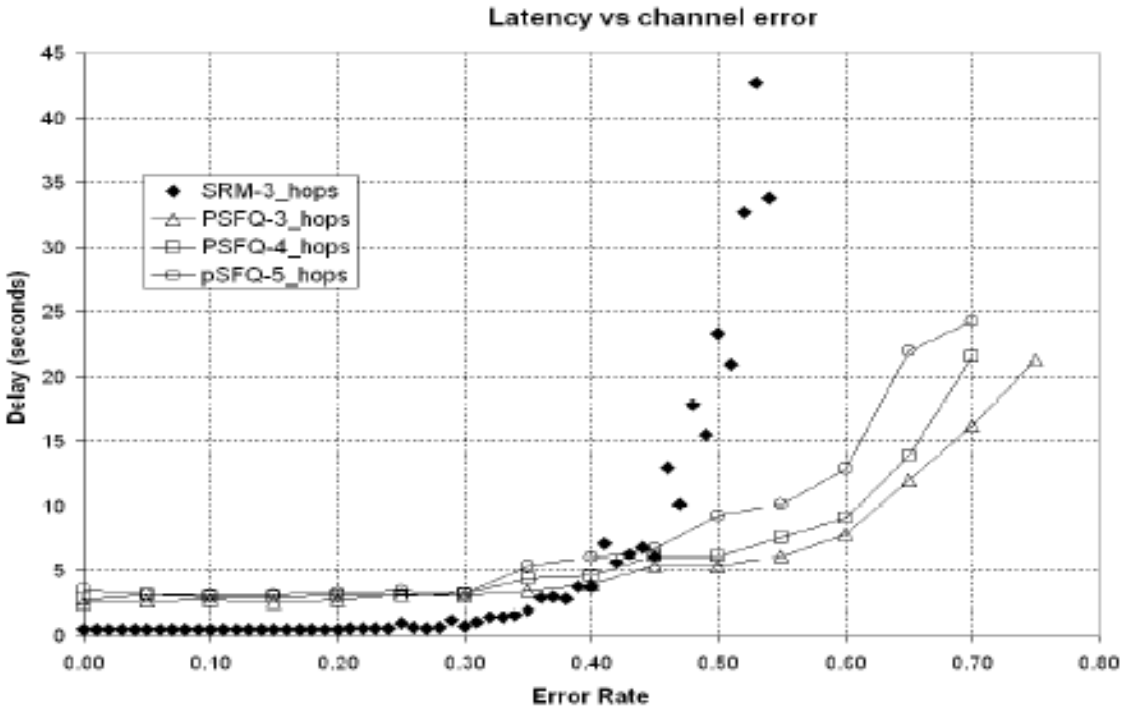


Figure 6. Comparison of average latency as a function of channel error rate.

PSFQ : Pump Slowly Fetch Quickly

- Simulation : results : 70% error rate op bound for PSFQ

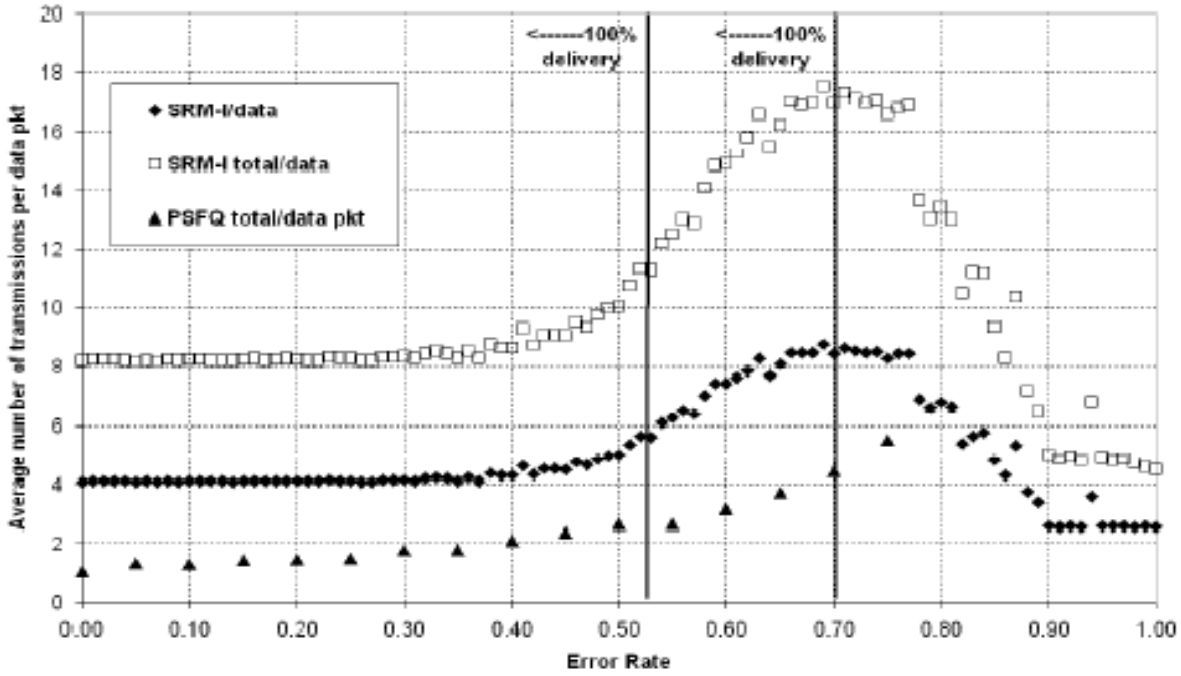


Figure 7. Average delivery overhead as a function of channel error rate

PSFQ : Pump Slowly Fetch Quickly

- Simulation : results
  - Average delivery doesn’t degrade with increasing hops at low error rates(30%).
  - At very high error rates(70%) average delivery ratio starts decreasing with increasing hops(>8).
  - Latency is constant till 30% error rates and slowly increases till 50% error rates and starts increasing after 50% error rates.
  - Retransmission overhead for recovery increases at 70% error rates. At error rates below 70% it increases very slowly.

PSFQ : Pump Slowly Fetch Quickly

- Simulation : Conclusions
  - High error tolerance.
  - Scalability – scales to large number of hops without much degradation.
  - Adaptive – Adapts retransmission rate, type of forwarding etc. based on error rate, loss factor, neighbors activity. (How many do not believe it is adaptive, How do we define adaptive in transport layer protocols ?)

PSFQ : Pump Slowly Fetch Quickly

- Discussion – Topics
  - What happens in Source to Sink communication when Length is not known or changes while communication. (Application handles this using control messages, Multiple windows of data, Flow associated Windows).
  - Handling Single Packet message lost (e.g. control or management message, assuming per flow sequence number)
  - State maintenance at Hops e.g. storing NACKs for Id's in Loss window by different nodes, storing transmitted packets from Loss window, maintaining NACK counts from 1 source and propagating in case no reply is heard.

PSFQ : Pump Slowly Fetch Quickly

- Discussion – Topics
  - What happens if pump fast is employed and stop pumping if retransmission requested ? (e.g. Analogical to TCP congestion control)
  - Effect of In-sequence forwarding. Possibility to optimally use out of sequence forwarding with pump fast. Analyze performance and cost.
  - Rate control at Source.
  - How PSFQ works with data dissemination

PSFQ : Pump Slowly Fetch Quickly

- Discussion – Topics
  - The maximum time length of stored packet can be Tmax * n-1 =Tpro if fwd communication is stopped. What happens if buffer at intermediate node is full. It may start dropping packets (older/recent). Proactive fetching by a destination node may not help. Reporting can be used.
  - Data Caching (cache size). How long to store in cache ? What happens when multiple application using the sensor network as relay ? What happens if Report is lost

PSFQ : Pump Slowly Fetch Quickly

- Discussion – Topics
  - Limited Broadcasting, Group/Cluster Based Communication.
  - Leveraging reliability of lower layers with closer integration at the same time keeping it customizable.