DATA MANAGEMENT IN WSN BY STRUCTURED REPLICATION

Prof Narendra Kumar
PhD Research Scholar, *Manav Bharti University, Solan, Himachal Pradesh (India)*

ABSTRACT

Wireless Sensor Networks (WSNs) have matured as the technology most suited for monitoring of the environment and data collection. But harnessing the powers of a WSN presents innumerable challenges. Issues related to networking (routing), storage, and transport of the huge amount of data that keep flooding the Network, needs to be planned in advance, before the data reaches the sink. All the foreseeable contingencies have to be taken into account to minimise creation of hotspots on the sensors around the sink. Some practical aspects are examined and useful methods of overcoming or at least circumventing the communication hotspots are described in thispaper.

Keywords :
Communication Hotspots. *The bottlenecks arising due to flooding of the data or failure/ tardiness of a link, path, hardware interface ofNetwork.*
Data Centric Storage (DCS) *Stores data in a subset of the sensors decoupling the mechanism of data production from datacollection.*
Sensor-net *- Distributed sensor-network of a large number of small devices.*
QoS (Quality of Service) *- Ensures data integrity and authenticity.*
Q-Night *- Rejection hashing technique that ensures QoS and load balance.*
Unicast Communications*- Allows in-network data pre-processing.*
GLS
Geographic Hash Table *- Geographically mapping a key–value pair and storing it in the vicinity of the location to which its key hashes [Put() and Get() operation done on the same key k that puts hash k in that verylocation].*

INTRODUCTION

For many of the applications using Data-Centric Storage (DCS) for location guidance, it is expedient to combine LS and DCS by storing detailed event information locally and using DCS to inform of an event's location to those raising query on it, so that subsequent queries can be directed to the proper place - somewhat like the ORACLE Database Management System. Geographic Hash Table (GHT) helps in keeping relevant information in adjoining Nodes to help expedite the search process.

The core step in GHT is the hashing of a key ***k*** into geographic coordinates. Both a **Put*()*** operation and a **Get*()*** operation on the same key ***k,*** *actually puts the* ***hashk***to that very location. A *key–value pair* is thus stored at a node in the vicinity of the location to which its key hashes. Choosing this node consistently is central to the mechanism of building a GHT. If we assume a perfectly static network topology and a network routing system that can deliver packets to the desired location/ positions, ***such a GHT*** will cause storage requestsand

queries ***for the same k*** to be *routed* **to the same node**, and will ***distribute*** the storage request and query load for **distinct k values *evenly*** across the **area covered** by this network.

The service provided by GHT is similar in character to those offered by other distributed hash table systems. However, in the case of WSN systems, much of the nuances of the GHT system-design ***emerges specifically*** to ensure robustness and scalability in the face of the **many failures** that plague such distributed systems. GHT makes use of a ***novel*** *perimeter refresh protocol* that ensures both ***persistence and consistency*** when nodes fail or shift their location while on the move. This protocol ***replicates stored data*** for ***key k*** at nodes around the location to which **k hashes**, and ensures that one node is chosen consistently as the *home node* for that *k*, so that **all**storage requests and **queries** for *k* can be routed to thatnode.

DCS FOR CONTEXT WITH DIRECTED QUERIES (GEOGRAPHICALLY TARGETED)

During the normal working environment most of ***the data-communication*** protocols and plans for storage of the huge amount of data emanating from various sensors work efficiently. These protocols typically use the local communication resources. Hash-keys/ GHT helps in keeping track of events but the system as a whole is beset with many problems that must be resolved for it to become efficient and effective.

Typically DCS in WSN poses following problems :

- *Storage abstraction.*
- Node failures.
- *Topology changes.*
- System scale in nodes.
- *Energy constraints.*
- Persistence.
- *Consistency.*
- Scaling in database size.
- *Scaling in node count.*
- Topological generality.

The DCS system architecture described in this paper ***helps to overcome or at least*** **alleviate** many of these problems by extensively using **GHT** *(a Geographic Hash Table). Some* ingenuity is required in sifting the data.

MANAGING THE FLOW OF DATA

The ***core step*** in GHT is the **hashing of a key** *k* into geographic ***coordinates***. Both a **Put***()* operation and a **Get***()* operation on the same **key *k***, hash *k*to the same location. The key–value pair so generated is stored at a node in the vicinity of the location to which it's key hashes. Choosing this node consistently is central to building a GHT. If we assume a perfectly static network topology and a network routing system that can deliver packetsto

positions, such a GHT will cause storage requests and queries for the same *k* ***to be routed to the same node***, and will distribute the *storage request* and *query load* for distinct *k* values evenly across the area covered by a network.

COMPARING GEOGRAPHIC WITH DISTRIBUTED HASH TABLE

The services provided by GHT, is similar in character, to those offered by other ***distributed hash table*** systems. However, as is the case with those systems, much of the nuances to the GHT system design arise specifically to ensure robustness and scalability in the face of the many facets of failures possible in a distributed system.

GHT – as the name suggests, utilises geographical proximity for regulating the data-flow. It does so by using a novel ***perimeter refresh protocol*** to provide both ***persistence*** and ***consistency*** when nodes fail or move. As this protocol **replicates** stored data for key *k*, at nodes around the location to which *k* hashes apart from ensuring that one node is chosen consistently as the *home node* for that *k*, so that all storage requests and queries for *k* can be routed to that node. It appears tardy but even for the networks where nodes are deployed densely, the protocol is efficient.

GHT typically relies on extensive use of local communication, especially on networks where nodes are deployed densely, to achieve high efficiency. By hashing keys, GHT **spreads** storage and communication load between different ***keys*** evenly **throughout** the sensornet. When many events with the same key are stored, GHT avoids creating a hotspot of communication and storage at their shared home node by employing ***structured replication***, whereby events that hash to the same home node, can be divided among multiple mirrors.

ALGORITHM

The algorithm that propel the GHT, is built atop GPSR, a *geographic routing system* for ***multi-hop*** wireless networks. As an important features of GPSR's design relevant to GHT, it is worthwhile to mention that this is a less-nexploited characteristic of GPSR but allows all packets destined for an arbitrary location (unoccupied by a node) to be routed consistently to the same node, in the vicinity of that location. GHT brilliantly leverages this characteristic to route storage requests and queries for the same key to the same node, despite the ignorance of the hash function that maps keys into locations of the placement of nodes in the network.

GPSR CHARACTERISTIC

Under GPSR, packets are routed geographically. All packets are **marked** with the *positions* of their destinations. All nodes know their own positions as also the positions of the nodes that are a single hop away from them. Using only this local knowledge, ***GPSR can route*** a packet to any connected destination. All that is then needed is to find the most suitable route to that Destination. Greedy algorithm does therest.

GREEDY FORWARDING EXAMPLE

Consider a simple Greedy forwarding example i.e. Node *x* forwards to *y*, its neighbor closest to *D as* x has no neighbour closer to *D*. The *greedy* forwarding algorithm goes on repeating this process, so as to move the packets progressively closer to the destination at each hop. In case of a dead end, the *perimeter* forwarding algorithm takes over to complete the task.

The greedy forwarding rule is simple: a node *x* forwards a packet to its neighbour *y* that is closest to the destination *D* marked in the packet, so long as that neighbour is closer to *D* than *x*. The logic being sound the Greedy forwarding algorithm fails when no neighbor is closer than *x* to thedestination.

WEEDING OUT REDUNDANCIES

Even when two paths exist from x to *D*, *x maynot* forward ***greedily*** on either of them, if both the axis involve temporarily moving the data *farther away* than *x* from the destination. In cases of an error, the good-old GPSR comes to the rescue. GPSR recovers from greedy forwarding failure using *perimeter mode*, which amounts to forwarding packets using the *right-hand rule*. Upon arriving on an edge at node *x*, the packet is forwarded on the next edge counter-clockwise about *x* from the ingress edge. This process causes packets to tour enclosed faces as shown intuitively, it is useful for circumnavigating regions where greedy forwardingfails.

ROUTE MANAGEMENT

GPSR routes ***perimeter mode*** packets on a planar subgraph of the network connectivity graph, in which there are no crossing of the edges. A perimeter is a face of this planar graph. There are many algorithm that uses planar network subgraphs to ***recover/ take-over*** on failure of greedy forwarding algorithm.

GPSR originates packets in greedy mode, but ***changes them to perimeter mode*** when no neighbor of the forwarding node is closer to the packet's destination than the forwarding node itself. GPSR returns a perimeter-mode packet to greedy mode when the packet reaches a node closer to the destination than that at which the packet entered perimeter mode (stored in the packet).

GPSR NETWORK MODEL

GHT algorithms use ***perimeter mode*** in another novel way, to route packets that refer to the *same* storage key to the *same* node. GPSR was designed for a network model where a sender wishes to transmit packets to a destination node with a known ***non-geographic address***. Sensors of a WSN are essentially that, so a sender maps the destination's identifier to its current location using a location database, such as ***GLS*** . Under GHT, the originator of a **Put***()* or **Get***()* packet does *not* know the identifier of the node that is the eventual destination ofthepacket.Theoriginatorofa**Put***()*or**Get***()*forakey*kjust*hashesthe**name*k***into

geographic **coordinates** that are the ***destination of the packet*** for that operation. The hash function is ignorant of the placement of individual nodes in the topology. It merely ***spreads*** the different key names ***evenly*** across the geographic region where the network is deployed.

GPSR'S PERIMETER MODE & THE HOME NODES OF GHT

No node may be available at the precise coordinates the hash function produces. The *home node* for a GHT packets, being the node ***geographically nearest*** the destination coordinates of the packet, serves as the ***rendezvous*** point for **Put***()* and **Get***()* operations on the same key. As GHT packets are not addressed to a ***specific node***, but only to a **specific location**, it is treated by GPSR as a packet bound for a disconnected destination. No receiver ever sees the packet addressed to its own identifier and the GPSR merely routes such a packet to the appropriate ***home***node.

GHT uses GPSR's perimeter mode to find these home nodes. Under GHT, the packet enters perimeter mode at the home node, as no neighbor of the home node can be closer to the destination. The packet then traverses the entire perimeter that *encloses* the destination, before returning to the home node and , the home node knows to consume the packet when it returns after this tour of the home perimeter. So the progress can be easily monitored and controlled.

SUMMARY

The maze of disjointed Wireless Sensors in a WSN may appear to be unmanageable from data-communication and storage point of view, but it presents an interesting challenge to the Data-Managers that is both enigmatic and This paper presented the design and evaluation of GHT based on DCS system for sensornets built on geographic routing. The simulation of networks having up to 100,000 nodes indicates that DCS offers reduced total network load and hotspots as compared with external storage and local storage. For the WSN of large node populations as well as where many events occur concurrently, the GHT leverages the GPSR geographic routing system to offer an efcient DCS service that maintains persistence of data when nodes fail or move. Also, the benefits accruing from such scheme include scalability achived by spreading the load of (key, value) pairs evenly throughout the sensornet.

Several avenues needs to be explored further. Firstly, the effect of varying node density. Under GHT. The keys being uniformly hashed over the geographic space, as the nodes are dispersed/distributed increasingly non-uniformly, the scheme of the storage-and-forwarding-of-load across WSN to be correspondingly lopsided. There is a need to anticipate the consequent performance implications and devise necessary mechanisms to cater to such non-uniformity as also to avoid hashing keys to points outside the sensornet,

REFERENCES

1. Geographic Routing for Wireless Networks by Dr B. Karp in his Ph.D. Dissertation, Division of Engineering and Applied Sciences, Harvard University, (Cambridge, Massachusetts, October 2000).

2. Routing *and Addressing Problems in Large Metropolitan-Scale Internetworks,* by G.G. Finn, USC/ISI Technical Report ISI/RR-87-180, (March 1987).

3. Sensor Information Technology, in Defense Advanced Research Projects Agency, http://www.darpa.mil/ito/research/sensit.

4. Wireless *Sensor Networks Simulator*, by McCanne and S. Floyd, http://www.isi.edu/nsnam/ns/.