Energy Efficient Data Management in Sensor Networks*

Contact Author: Sanjay K Madria, Dept of CS, Missouri S & T, Rolla, MO

First, I provide a brief overview of our most recent work done in this area and then discuss further research issues.

**Energy Efficient Secure Data Aggregation in WSN [1]:** A wireless sensor network (WSN) is a spatially distributed and self organizing network of a large number of low cost sensors. Wireless sensors are typically battery operated devices, which implies that they have limited energy. Since in most cases, it is not feasible to replace exhausted batteries once the network is deployed, we try to extend the network life time by making judicious use of the limited power available. A sensor expends most of its energy in communicating with other sensors. Keeping this in mind, our objective is to minimize the communication and computational energy in a sensor network in order to increase the sensor life time. Data aggregation is an efficient way to minimize energy consumption on sensors, but it introduces new security challenges. Hop by hop security is no more an option in this case as it entails decrypting the incoming data in order to perform aggregation which has two hazards associated with it. First, it makes the aggregator loose energy faster and second, it potentially exposes the raw data to the adversary. To make data aggregation secure, we need an end to end scheme, where the data is encrypted and signed at its source and decrypted and verified only at a single trusted place (usually the base station). In addition to this we also need to make sure that we don’t burden the nodes with complex computations and needless transmissions which will make them loose power quickly. Our objectives are as follows.

1. An encryption algorithm for data confidentiality which will allow us to aggregate encrypted data in a wireless sensor network.
2. An aggregate digital signature algorithm to preserve data integrity which allows us to aggregate digital signatures.
3. A WSN test-bed incorporating these algorithms efficient in computational and communicational aspects in energy constrained environments has been demonstrated using real sensor motes.

**Energy Data Compression in WSN [2]:** We proposed TinyPack, a compression scheme for real-time sensor networks. TinyPack reduces the amount of data flowing through the network without introducing delays. First the data is transformed by expressing the sensed values as the change in value from the previous sensed reading. This is referred to as delta compression. We demonstrated its effectiveness for any generic real-time sampled dataset. Second, the individual delta values are then compressed using a derivative of Huffman coding. Huffman codes express more frequent data values with shorter bit sequences and less frequent values with longer ones. The codes are generated and updated dynamically so no delay is needed. TinyPack is a lossless compression algorithm and the data can be decompressed at the sink or base station without any loss of granularity or accuracy. It outperforms current work in this area.

**Energy Efficient, Adaptive and Collaborative Data Sampling in Wireless Sensor Networks:** In distributed data stream networks many sensors continuously deliver data to a central base station. As the networks become larger, it becomes important to effectively allocate and conserve bandwidth. The naive approach would be to simply allocate the same amount of bandwidth to every sensor at all times, but many applications exist for which this is unrealistic and ineffective. For example, in troop tracking, units which are near enemy territory or units containing important personnel and equipment should receive higher priority and more attention than units that are stationary or in relatively safe areas. Similarly, in environment monitoring applications where sensors measuring temperature, light, and humidity were deployed in a laboratory, sensors near doorways or windows could see a higher variance in data. It would then be useful for these sensors to have a higher sampling rate or more bandwidth. Sampling rates for all sensors could also be decreased at night when the lab was empty and few environmental changes were occurring to conserve sensor nodes energy.
Centralized approaches to dynamic sampling rates exist in which the central base station takes requests from sensors for higher sampling rates and allocates more bandwidth to the sensors in determines to be more important. This, however, can put an undue burden on the base station. We propose a distributed solution in which neighboring sensors collaboratively adjust their sample rates in order to maximize the flow of important information through the network and at the same time also balance energy consumption.

**General Approach:** We consider an arbitrary system where sensors send data streams through the network to a base station either directly (one-hop) or through aggregating and forwarding nodes (multi-hop). We define each set of sensors which send to the same entity to be a cluster. The sensor or base station they send to is defined as the cluster head. (Note: in a multi-hop network each intermediate cluster head is also a peer to sensors in another cluster). Within a cluster, sensors which are within radio range of each other are considered neighbors. The system uses time division multiplexing where the cluster heads allocate equal time slots to each of the other nodes in the cluster.

When a sensor requires a higher sampling rate then it has currently been allocated, it listens to its neighbors’ transmissions and sends a request for assistance to the neighbor which is not utilizing the largest chunk of its time allocation. The neighbor then will allow the burdened sensor to use a portion from the beginning of its time slice to send data. If a sensor has more data than can be sent in its current timeslice it sends as much as it can using compression to increase the amount of data that can be sent.

**Time Synchronization:** Since the system relies on all the sensors sending at specific time intervals it will be necessary to use some form of time synchronization. Global synchronization is not required. Merely keeping time synchronized within a cluster would suffice for this system. A simple, yet effective method would be for the cluster head to monitor what time all the packets are coming into and send a response to any sensor which has fallen out of synch reporting the amount of clock skew it is exhibiting. The sensor can then correct itself to become synchronized to the cluster head.

**Graceful Failures:** If a sensor begins continually falling well out of sync, that sensor is most likely running out of power and beginning to fail. Therefore if a sensor begins exhibiting this behavior, it can assume it is failing and remove itself from the network. It then gives all of its timeslices to sensors with adjacent timeslices for them to distribute in the normal fashion.

**Handling Excess Load:** If no sensor is available to service the request for more bandwidth, the overloaded sensor will buffer its data until its next timeslice. If the buffer becomes too full, the load will have to be shed. A sensor can simply shed the oldest data and replace the data in the buffer with the newest data. Another method would be for the sensors to collaboratively determine which sensor should shed data.

**Handling Collisions:** In the case where two sensors become overburdened simultaneously, both will attempt to request more bandwidth at the same time and may result in a collision. Since the requests for more bandwidth will typically be small compared to the actual data, the time for sending the request can be chosen randomly or based on the id of the sensor to reduce the probability of collision. If a request takes 1 ms to send and the sensor receiving the request is currently not using 12 ms of its timeslice the requesting sensor can send the request (requesterID mod 12) ms after the destination sensor has finished sending its data which would reduce the chance of collision by a factor of 12. If a burdened sensor hears another sensor requesting more bandwidth, it will defer to that sensor and request bandwidth from the next unburdened sensor. If two requests are sent to the same sensor, the allocation is given to the sensor with the greatest need. Ties are broken by sensor ID.


* supported by grants from Army Research & DOE and ideas are also contributed by several students working for their PhD theses