SELF-COMMUNICATING AND SELF-CONFIGURING WIRELESS SENSOR ENVIRONMENT FOR INDEPENDENT LIVING SUPPORT

Bozena Kaminska[1], Pavel Gburzynski[2]
[1]*Simon Fraser University,* [2]*University of Alberta*

INTRODUCTION

Greater artificial intelligence and increased independence from human intervention are the key attributes of the wireless sensor environment for independent living support and for health monitoring. Wireless Sensor Networks (WSN) create an environment with embedded and hidden technology for measurement, monitoring and management of data from multiple sources and probes with little constraint on location. The enabling technology, comprised of the sensor nodes and the network infrastructure, requires employing advanced techniques in various areas such as MEMS sensors, microprocessors, memory, RF transceivers, smart antennas, Internet routers, etc [1]-[5]. The wireless sensors also demand new design concepts to satisfy the design requirements such as compactness, stability of the signal prone to motion and other disturbances, durability and ultra-low power consumption. Wearable Sensor Networks impose even more constraints such as long-term wearability with comfort, simplicity to use, reliable sensor attachment and fault tolerant functioning [3]. The target requirement is to deliver **reliable and sustainable operation** of a WSN. The factors of environmental conditions, changing behavior and perturbations need to be part of the definition of a WSN architecture and mode of operation.

A sensor network for health monitoring applications must balance the conflicting measures of cost, reliability and convenience. While the ad-hoc approach to organizing such networks is attractive from the viewpoint of reachability, flexibility and cost (because of their independence of infrastructure), the popular ad-hoc forwarding schemes promoted in the commercial world raise doubts regarding their resilience, robustness and reliability. The very term "ad-hoc" suggests sloppiness in service: few people would be comfortable betting their lives on an ad-hoc health monitoring system.

We present a certain generic "ad-hoc" networking solution to low-cost, reliable and non-intrusive health monitoring, whereby the communication paths carrying the critical sensor data are automatically safeguarded against failures or individual nodes and provide stable connectivity in the face of node movement or blackout. This is in contrast to most of the commercially available systems (e.g., ZigBee®) where lost paths must be recovered from, which necessarily introduces episodes of uncertainty and unpredictability into the network behavior. With our solution, under reasonable conditions of general connectivity (meaning that the network is not fundamentally disconnected) paths are never lost and there is nothing to recover from. This means that the streams of data flowing from (and to) the sensors are never disrupted. Node mobility and occasional local failures or blackouts need not cause hiccups in the delivery of critical data.

CLASSICAL AD-HOC NETWORKING

The prevailing wisdom regarding the organization of an ad-hoc wireless network assumes point-to-point communication, whereby each node forwarding the packet on its way to the destination sends it to a specific neighbor. Popular routing protocols can be broadly divided into two groups. Proactive protocols try to maintain up-to-date routing information at every node in anticipation of demand [*6*], while reactive protocols collect the necessary information only when it is explicitly needed to sustain an actual session [*7*]. Regardless of the assumed paradigm, the bottom line is pretty much the same: the primary objective of the routing scheme is to determine the exact sequence of nodes to forward the packets from point A to point B.

Let us consider AODV [6] as an example. In several marginally different guises, this is one of the most popular ad-hoc routing schemes and, in particular, it lies at the heart of ZigBee. A node *S* initiating packet exchange with node *D* broadcasts a request to its one-hop neighbors to start the so-called *path discovery* operation. Based on its current perception of the neighborhood (neighborhoods are constantly monitored by all nodes) and cached information collected form previous path discoveries, a node receiving such a request may decide to forward it elsewhere, or, possibly, respond backward with a path information intended for the initiating node *S*. Depending on the relative stability of neighborhoods (i.e., mobility and failures of nodes) and the availability of alive cached routes, the path setup bureaucracy

may take more or less time. At the end, a single path between *S* and *D* has been established. This means that, when dispatching a packet to *D*, node *S* knows precisely which of its immediate neighbors the packet should be addressed to. Then, every node receiving such a packet knows the exact identity of the single next hop neighbor, and so on. *A problem arises when the path is broken because such a mishap effectively demolishes the entire delicate structure.* When that happens, a new path recovery operation is essentially started from scratch.

DO WE NEED ROUTES AT ALL?

Our ad-hoc forwarding scheme abolishes point-to-point forwarding. Note that in the wireless environment all transmissions are necessarily broadcast, and addressing a packet to a single neighbor does not help the fact that all other neighbors can hear it as well. Traditional schemes view this feature as a rather serious problem and try to defeat its negative consequences (hidden/exposed terminals via MAC-level tricks intended to facilitate point-to-point transmission. As we argue elsewhere [*4*], the merits of such tricks are questionable in sensor networking, where individual packets tend to be short. Our opinion is that instead of fighting the inherent broadcast nature of a wireless channel, we should embrace it as a feature.

Suppose that node *S* wants to send a packet to node *D*. With our scheme, *S* simply transmits (broadcasts) the packet to its neighbors. A neighbor may decide to drop the packet (if it believes that its contribution to the communal forwarding task will not help) or retransmit it. This process continues until the packet reaches the destination *D*. An important property of this simple generic scheme (otherwise known as flooding) is that a retransmitted packet is never specifically addressed to a single next-hop neighbor. At each hop, the packet is in fact re-broadcast. There is nothing wrong in the fact that multiple neighbors can hear it at the same time. This actually works to our advantage.

Needless to say, to make such a scheme useful, measures must be taken to limit the number of retransmissions to the minimum at which the desired quality of service is still maintained. This part comes as a series of *rules* that determine when a node receiving a packet should rebroadcast it, as opposed to dropping. Some ideas for such rules are obvious, e.g., discarding duplicates of already received packets, limiting the maximum number of hops traveled by a packet. The key to the success of our variant of flooding is a rule that brings the paths traveled by forwarded packets down to a narrow (but intentionally fuzzy) stripe of nodes along the best route.

Consider three nodes *S*, *D* and *K* depicted in Figure **1**. *K* is contemplating whether an intercepted packet sent by *S* and addressed to *D* should be retransmitted or dropped. Its decision is based on simple calculations involving certain parameters cached by *K* and extracted from the packet headers.

We assume a bi-directional nature of data exchange, which means that a stream of data from *S* to *D* implies an associated (possibly much less intense) stream of data from *D* to *S*. Nodes learn about the sessions currently present in the network by monitoring packets passing by and collecting relevant information from their headers. Until the network learns about a particular session (understood as a pair of nodes that want to communicate), the forwarding for that session may be overly redundant. However, with time, the route will tend to converge to "best" path. Note that this mode of operation is not fundamentally less economical than one based on point-to-point forwarding (in a traditional ad-hoc routing scheme). This is because before a point-to-point system can commence forwarding it has to "learn" the routes one way or the other. No information comes free out of the blue. In a traditional scheme, there are distinct phases of knowledge acquisition and recovery from its decay, which are separate from the actual forwarding. In our approach, the acquisition and maintenance of the requisite knowledge happens together with forwarding (as its byproduct) and constantly acts to preserve and improve its quality.

In addition to *S* and *D*, each packet traveling from *S* to *D* (as seen by *K*) carries the following information: h_f – the (forward) number of hops traveled by the packet so far and h_b – the (backward) target number of hops seen at the source *S* for a previous packet arriving from the opposite direction, i.e., from *D*. The latter parameter is acquired by *S* when it receives a packet from *D*. Note that as duplicates of already received packets are identified and promptly discarded, the first new packet that makes it to the destination usually has traveled along the shortest (or fastest) route between the two endpoints.

CAPITALIZING ON THE SUBOPTIMAL PATH DISCARD RULE

Owing to the inherent imperfections of the ad-hoc wireless environment, *K* should not be too jumpy with negative decisions. One possibility is to include a *slack* parameter *m* in the inequality (as shown in Figure **1**). Note that when $m > 0$, the rule will allow the node to forward the packet when the path passing to it appears to be slightly longer (by up to *m* hops) than the currently believed shortest path. In essence, the slack parameter *m* determines the degree of fuzziness (or redundancy) in the population of nodes separating

S and *D* and deemed responsible for packet delivery. Note that we never mention the concept of a *route* understood as an exact hop-by-hop prescription for getting a packet from *S* to *D* (or back). When a packet departs at one end of the session, it doesn't know the exact path that it will travel. All it knows is that a certain (possibly fuzzy) subset of nodes will collaborate in the communal task of its delivery. The identity of those nodes is irrelevant to the point that they don't even have to be addressable! Indeed, an internal node that never originates nor absorbs traffic needs no address.

The scheme can be augmented by simple heuristics that probe the network for new routing opportunities. For example, no matter how unsuitable a node appears to forward a packet, it may decide to do so every once in a while – just to check if the packet cannot be (better) delivered via an alternative segment of the network. Recoveries from natural lapses resulting from momentary local blackouts or reasonable mobility of nodes can be accomplished in a completely non-disruptive manner with the proper setting of the slack parameter *m*.

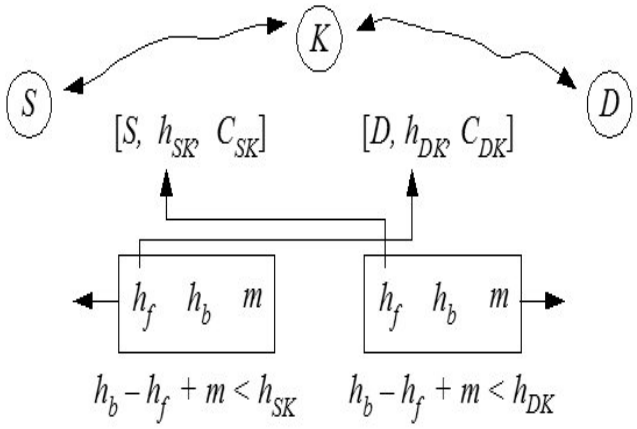

Figure 1: The suboptimal path discard rule.

APPLICATION TO SENSOR NETWORKS

The presented scheme constitutes the basis of TARP [*1]* – our comprehensive add-hoc routing protocol for small-footprint, reliable, self-healing and inexpensive mesh networks catering to sensing applications. A typical configuration of such a network involves one or more collection points interspersed among a dynamic and possibly mobile collection of sensing nodes, as shown in Figure 2. A collection point can be viewed as regular node (as it has to communicate with sensor nodes over the wireless channel) additionally equipped with an OSS interface.

Although typically most traffic in such a network flows from sensors to collection points, there is usually some demand for traffic in the other direction – to issue commands to the sensor nodes (switching them on and off, selecting data to retrieve) or notifying them about the collection points that they should report to. In any case, the equivalents of sessions tend to have a strongly asymmetric nature: each of the collection points usually maintains a large number of sessions (with all their reporting sensor nodes) while a sensor node typically cares about a single session – with its target collection point. This asymmetry is OK with TARP. As long as a reasonable amount of traffic flows in either direction, the nodes along the way are able to fill their caches and take advantage of the rule illustrated in
Figure **1**. Typically, the collection points broadcast periodic beacons that fulfill a dual role: providing the heart beat and advertising the collection points to the sensing nodes, as well as filling their caches with the current value of h_{SK}, i.e., the number of hops separating them from the collection point.

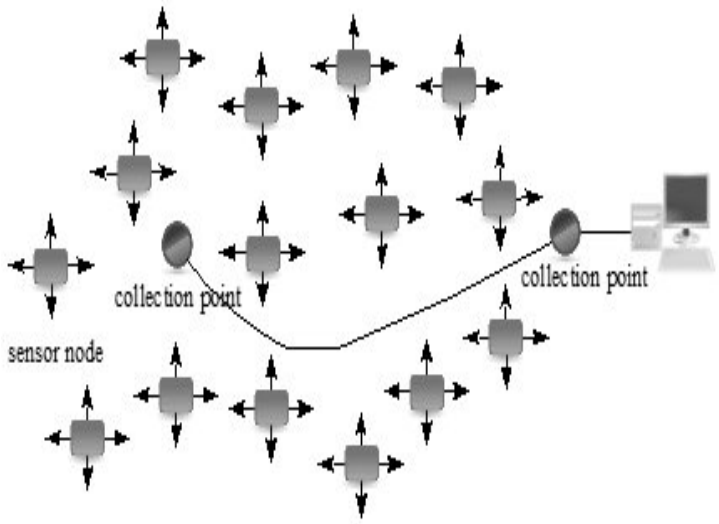

Figure 2: A sensor network.

SMOOTH HANDS-OFF [1]

The setting of the slack parameter *m* trades off the global bandwidth for reliability and smoothness of operation in the face of possible reconfigurations of the mesh. It usually makes sense to set this parameter in such a way that the paths are maximally redundant

[1] We do not know the neighbourhoods of nodes *A* and *C* because we ignore whatever happens outside the cloud. Consequently, we do not consider their disappearance.

while the network can still comfortably deliver all sensed data on time without congestion. It is possible to use feedback from collection points (or even regular nodes) in adjusting *m* for best performance.

To see how the network copes with mobility and node failures, consider the scenario shown in Figure 3. Packets traveling between nodes *U* and *V* are forwarded within the clouded fragment of the mesh network. For simplicity, we ignore whatever happens outside the cloud (the presence of other nodes along its boundary can only help). Suppose that the arrows represent neighborhoods, i.e, single-hop reachability. The path *A*-*B*-*C* (of length 2) is the shortest route through the cloud, thus, it will be considered the "yardstick" to which other nodes overhearing the packets forwarded by *A*, *B*, and *C* will calibrate their decisions. Suppose that *m* is modestly set to 1. This means that nodes *E* and *F* will also retransmit the packets because the route through them incurs a 1-hop increase over the best path. The worst that can possibly happen is the disappearance of node *B*, which is a critical component of the current best path.[2] Note, however, that this disappearance will not disrupt the traffic because the second best path through the cloud, i.e., *A*-*E*-*F*-*C*, is also being used. The net outcome of this disappearance will be that a would-be duplicate arriving at *A* or *C* (from *E* or *F*), that would be discarded and ignored if *B* were in place, will be bona-fide received and forwarded towards the destination. After a short while, as the destinations update their h_b values in response to the increased number of hops along the best path, the nodes within the cloud will learn that the *A*-*E*-*F*-*C* path is the best one[3] at the moment. Subsequently, nodes *D* and *G* will add themselves to the population of forwarding nodes, as the path through them is only one hop worse than the current minimum. As we can see, assuming the changes are not too rapid or too drastic, they are accommodated without any disruption in service whatsoever. Note that with a higher setting of *m*, the network would be resistant to more drastic changes, at the cost of increased redundancy.

An important property of that redundancy is its controllability. Note that *m* can be viewed as a dynamic parameter, adjustable by the nodes, whose value reflects the momentary priority of reliability over bandwidth or vice versa. Many sensor networks need little bandwidth. For them, *m* can be set high, still providing a methodologically motivated alternative to naïve and uncontrollably wasteful flooding.

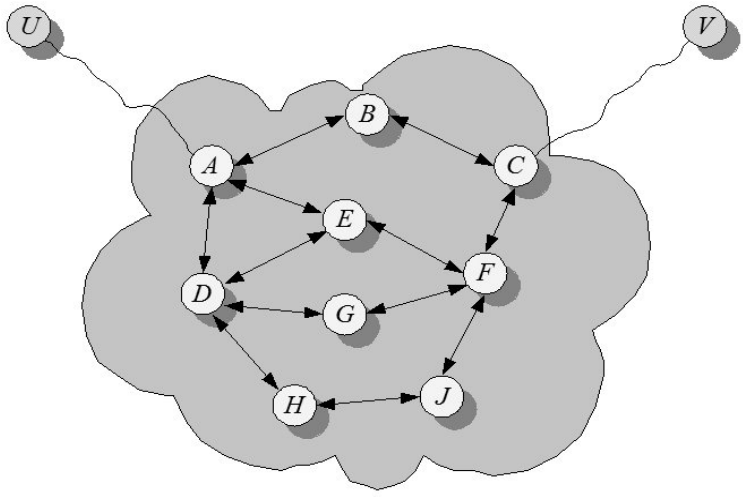

Figure 3: Illustration of hand-off.

CONCLUSIONS

The presented wireless sensor environment has been implemented for home monitoring of daily activities and physiological parameters (ECG). For daily activity monitoring the accelerometers and movement sensors have been used. During few months of operation none of 4 houses and subjects presented any problem with movement, reliability or timely information. Any non-functional sensor node (communication problem, battery, confusing data) was eliminated from the operation. The system could maintain full functionality. The statistical data and comparisons will be presented elsewhere.

REFERENCES

[1] Gburzynski, P., Kaminska, B. and Olesinski, W., "A Tiny and Efficient Wireless Ad-hoc Protocol for Low-cost Sensor Networks", DATE 2007.

[2] Kaminska, B. and New, W., "Wearable Biomonitors with Wireless Network Communication", IEEE Engineering in Medicine and Biology, EMBS 2005.

[3] Kaminska, B. "Wireless Micro and Nano Sensors for Physiological and Environmental Monitoring", NSTI Nanotech 2006, Boston, May 2006.

[4] Akhmetshina, E., Gburzynski, P., and F. Vizeacoumar, "Picos: a tiny operating system", Proc. ESA'03, pp. 116-122.

[5] Gburzynski, P., Kaminska, B.,"Enhanced Dominant Pruning-based Broadcasting in Untrusted Ad-hoc Wireless Networks", IEEE International Conference on Communications, ICC 2007, June 2007.

[6] Lou, W. and Wu, J.,"On reducing broadcast redundancy in ad–hoc wireless networks", *IEEE Trans. on Mobile Computing*, 1(2):111–123, 2002.

[7] Stojmenovic, I., Seddigh, M. and Zunic J., "Dominating sets and neighbor networking protocol. *IEEE Trans. on Parallel and Distributed Systems*,13(1): 14–25, January 2002.

[2] Strictly speaking, they will not learn that this specific path is the best one, only that its length is the currently achievable minimum.