Proactive *vs* reactive routing for wireless sensor networks

Alexandros Koliousis and Joseph Sventek

Department of Computing Science, University of Glasgow,
17 Lilybank Gardens, Glasgow G12 8RZ, United Kingdom
{koliousa,joe}@dcs.gla.ac.uk

Abstract. Sensor network routing protocols must ensure the stability of the network infrastructure under varying network dynamics. Recovery from changes or failures is necessary to guarantee the availability of collection or dissemination paths. Routing protocols may react to path requests or proactively maintain a connected graph. Although functionally equivalent, proactive and reactive protocols are associated with different costs, in terms of resource overhead (e.g. energy or bandwidth) and non-functional guarantees (e.g. end-to-end delay, or time to repair). The protocol of choice must satisfy the system objectives, yet it should not require excessive resources. We study the behavior of proactive and reactive routing schemes, using prototype TinyOS implementations of the OSPF and AODV protocols, respectively. Simulation results show that there exist regions in the time-to-repair $\times$ overhead space where reactive protocols are preferred over proactive and vice versa; the proactive preference region grows as the number of simultaneous flows increases.

Key words: proactive routing, reactive routing, sensor networks

1 Introduction

Wireless sensor networks enable the *in situ* monitoring of physical phenomena in a broad range of environmental and habitat applications. Whilst constrained in energy, computing, memory, and bandwidth capacity, sensors can accomplish complex tasks in the aggregate [1]. Deployed in remote (usually inaccessible) areas, unattended sensors collaborate towards a common goal, as dictated by the application functional and non-functional requirements.

A sensor system is a large-scale distributed system. N sensors probe an area of interest for ambient phenomena and transmit their measurements, via multiple wireless links, to an authorized access point, usually a sink. As information traverses the network, intermediate nodes relay packets over an optimal path to the sink; optimality favors those paths that meet the required delivery guarantees. Sensors execute a distributed algorithm to produce a (common) routing table. The choice of routing algorithm is important to minimize resource usage in this constrained environment.

Sensor networks are error-prone. A subset of nodes will frequently encounter component or communications failures. Routing protocols must resolve these errors unaided by human operators while minimizing the failure duration – from the time a node detects a failure until all active paths converge to a stable state. Wireless links suffer transient periods of disconnection. Such transient failures occur frequently due to system errors (e.g. unsynchronized sleep periods) or external stimuli (e.g. physical obstacles in a path). Aggressive behavior to heal faults that are manifestations of transient errors can cause network instability. Thus, fine-tuning of protocol configuration parameters is essential to ensure an acceptable level of fidelity.

Sensors establish and maintain routes either proactively or reactively. Proactive protocols periodically monitor peer connectivity to ensure the ready availability of any path amongst active nodes. Sensors advertise their routing state to the entire network to maintain a common (partially) complete topology of the network. On the other hand, reactive protocols establish paths only upon request, e.g. in response to a query, or an event; meanwhile, sensors remain idle in terms of routing behavior. Sensors forward each routing request to peers until it arrives at a sink; the latter will respond over the reverse communication path.

Reactive routing protocols have been the protocols of choice in mobile *ad hoc* networks, due to frequent node mobility. Due to their simplicity, and inherent support for data on demand, they have been the predominant design choice in wireless sensor networks. However, uses of sensor networks differ from their mobile counterparts. For example, environmental monitoring involves stationary sensors collecting readings over time from fixed points in space [2], in contrast to hand-held devices of mobile users. We wish to determine if these different use cases make proactive routing more attractive in sensor systems. The sheer number of proactive and reactive routing protocols described in the literature mandates the classification of protocols into equivalence classes, and experimentation with a representative from each equivalence class.

We implement an exemplar protocol from the proactive and reactive equivalence classes, the Open Shortest Path First (OSPF) and the Ad hoc On-demand Distance Vector (AODV) protocols, respectively, in the TinyOS environment and examine their recovery process under varying network and protocol configurations. In particular, we measure the time to recover from loss of a sensor and the incremental communication overhead to achieve recovery. The results show that, given a network instance, there is a regime in the time-to-repair $\times$ overhead space where proactive routing is favored over reactive; this regime grows as the number of simultaneous flows increases. Our model, based on a vector-valued evaluation function, can generalize to multiple objectives and constraints in the requirements space.

2 Routing in wireless sensor networks

An integral part of a sensor system is a communication protocol for the reliable collection and dissemination of data. Sensor networks rely on *soft* or *hard* routing state to self-organize into tree structures. Soft state refresh timeouts can

implicitly detect a non-operational link in the network. However, it can take a substantial amount of time before sensors decide a destination is unreachable. To improve convergence in time-critical systems, sensors trigger hard state updates to explicitly notify their peers. Similarly, sensors utilize soft and hard routing state for leader election in hierarchical routing paradigms [3, 4].

Optimality in sensor routing protocols refers to resource awareness. A hop count metric can reduce the number of transmissions, but the marginal distance between sensors or the presence of noise may decrease the probability of delivery. Energy or bandwidth efficiency, in contrast to pure hop count metrics, can deliver service guarantees to sensor monitoring systems. For example, energy awareness diverts traffic from depleted paths, thus prolonging the system lifetime [5]; link quality measurements, e.g. received signal strength or packet delivery ratio, cause avoidance of lossy radio links, thus reducing transient failures [6]; packet load metrics cause distribution of application traffic for congestion control [7].

Routing algorithms calculate the best path per destination in a *distance vector* or *link-state* basis. In a distance vector protocol, optimality is computed incrementally along a path. Sensors calculate routes locally, based on their current, partial network state. They iteratively notify their neighbors of intermediate results until routing tables stabilize – if a different best path exists, they employ it. In link-state protocols, on the other hand, every sensor contributes to establish a replicated distributed database of the network topology. Then, sensors run a shortest path algorithm (e.g. Dijkstra's algorithm) over this topology database instance.

Distance vector protocols appear suitable for wireless sensor networks. First, sensor systems react to environmental stimuli, thus they acquire paths on demand. Second, the in-network processing model entails an incremental packet delivery. Finally, sensors usually propagate their measurements to one aggregation point; a single distributed tree structure is sufficient to support this many-to-one traffic model. As a result of these characteristics, the trend in sensor network routing has been towards distance vector algorithms, augmented by query-driven application models.

The replicated database model appeals to stationary sensor systems. By default, link-state protocols are more robust to network changes: they have sufficient state to route packets around broken links. Furthermore, sensors are usually immobile, or they move infrequently. Thus, they do not experience the thrashing of routing tables seen in mobile *ad hoc* networks. Management traffic is often overlooked; sensors may also accept re-configuration commands, software updates, or complete binary images from an authority external to the network, e.g. when an environmental scientist refines the in-network processing model to correlate new events). Given one or more access points, proactive protocols can promptly address an arbitrary set of nodes for administrative tasks.

The spatial relationship of events has led to data-centric routing models. In contrast to address-centric protocols, sensors direct packets according to data values or attributes, rather than node identifiers [8]. This application awareness of the communication stack matches the data aggregation paradigm in sensor

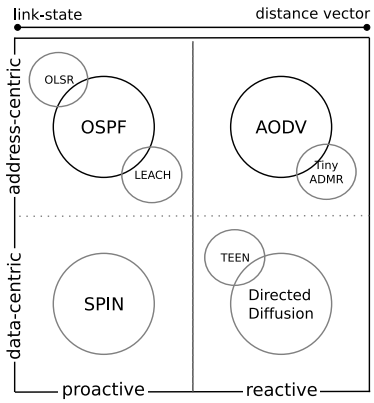


Fig. 1. Exemplar routing protocols for wireless sensor networks.

networks. The goal is to minimize transmissions by eliminating data redundancy, if present, during the collection process. In a similar manner, location-aware protocols utilize the relationship between sensors and the physical world to propagate packets according to absolute, or relative geographic coordinate systems.

Our classification of proactive and reactive protocols is orthogonal to address-centric and data-centric models (Figure 1). For example, in the SPIN family of protocols, sensors proactively advertise their measurements, via a 3-stage handshake protocol, to disseminate them across the network [9]. This gossiping is similar to a database exchange in address-centric proactive protocols. Directed Diffusion [10] borrows from *ad hoc* reactive protocols: a data query, or interest, determines the flow of data from one or more source nodes to a sink. Note that, although data-centric protocols change the routing semantics, the underlying routing mechanisms – the focus of this study – remain the same.

The two basic mechanisms of a routing protocol are neighbor discovery, to discover and maintain connectivity with peers, and flooding, to disseminate network state to distant nodes. Peer sensors exchange messages periodically to assert link liveness locally, within radio range. They flood local state to assert global knowledge of the network topology. Sensors utilize this local and global routing state to decide on best paths. The routing state traverses the network as control packets. There is a constant tension between the energy, memory, and bandwidth constraints within a sensor and the amount of routing state maintained by the protocol; the efficiency of the routing protocol is improved if more state is maintained, at the expense of increased utilization of the system resources.

2.1 The importance of neighbor maintenance

Sensors are n-resilient, where n is the number of their neighbors. That is a sensor can sustain $n - 1$ failures in close proximity and still actively participate in the system; it requires at least one neighbor to connect to the network. Sensors utilize soft state refresh intervals, or *hello intervals*, to ensure the fidelity of a link by listening to periodic broadcasts from peer nodes. Absence of control or

application traffic within that period indicates either a transient or permanent impairment in the network; sensors will react accordingly to diagnose and, possibly, resolve the problem. Neighbor monitoring can increase the confidence in fault diagnosis, and thus reduce the percentage of false alarms.

Broadcast packets can carry essential information to compute the desired link quality metrics between peer sensors, either implicitly or explicitly. For example, given a message frequency, a sensor can accurately derive delivery metrics, e.g. path delivery ratio or packet loss; TinyOS protocols reserve bytes to explicitly notify peers of radio quality, e.g. signal strength, RSSI, or transmission power. Beyond link metrics, sensors can periodically advertise data, or events, to direct data-centric flooding algorithms. Localization protocols also rely on neighbor state to converge to a relative or absolute coordinate space in the sensing area.

2.2 Broadcast forwarding

Flooding informs distant sensors of possible changes or updates in the network, e.g. a node failure or addition, or path requests, e.g. a query or event interest. Frequent floods will result in more accurate routing state amongst nodes. However, flooding poses a scalability problem, especially in dense networks, primarily due to excessive resource wastage. Routing principles for wireless sensor networks utilize local or global state to eliminate broadcast storms during flooding.

In simple flooding, sensors flood packets to all their neighbors. They maintain memory of their previous transmission to suppress any message duplicates. Essentially, sensors propagate packets only if they haven't previously sent a similar transmission. That is, in a network of n sensors, the process requires n transmissions to flood an update. Besides its simplicity, simple flooding is robust. Due to packet redundancy, it guarantees delivery in the face of network errors. For instance, flooding over a minimum spanning tree can reduce broadcasts, however a network error over this tree instance will disrupt the flooding process.

Simple flooding is the baseline system for sensor routing protocols: in the absence of network state, all protocols will perform similarly. For example in Directed Diffusion, unless sensors establish gradients, they will diffuse directionless packets. In order to eliminate redundancy, a protocol can enforce an upper re-transmission bound, or Time To Live (TTL), on packets assuming that a sink is reachable in TTL hops. Gossiping protocols [11] avoid broadcast storms by breaking synchrony amongst sensors; sensors remain silent for a random time interval before forwarding a packet listening for similar broadcasts and, if heard, they suppress it.

3 Exemplar routing protocols

We discuss the functional characteristics of two canonical examples of proactive and reactive routing, the Open Shortest Path First (OSPF) [12] and the Ad-hoc On-demand Distance Vector (AODV) [13] protocols.

OSPF is a proactive link-state protocol for routing within autonomous systems. Sensor systems are by necessity autonomous; we wish to explore the applicability of this wireline protocol in the realm of sensor networks. The idea is

to make OSPF aware of the sensor constraints, while maintaining its original features that made it attractive originally, viz. careful database synchronization and reliable flooding.

AODV is a reactive routing protocol for mobile *ad hoc* networks. It is its configuration parameters (i.e. soft state refresh intervals) that address node mobility. That is, the protocol should converge to steady state in zero mobility use cases. We revisit it because sensor networks borrow from reactive ad-hoc routing protocols in support of query-driven application models. Furthermore, recent monitoring systems relax the assumption of stationary sensors by incorporating support for mobile sensors (e.g. wildlife monitoring).

We implement the two protocols in the TinyOS 2.x environment to integrate issues of resource constraints and realistic radio models in our design. Our primary goal is to apply experiences from wireline and *ad hoc* routing to sensor systems. OSPF and AODV are sophisticated protocols; we refer the reader to the corresponding Request For Comments (RFC) documents for a complete specification of their operation. Below we discuss their basic characteristics, and appropriate modifications to adapt to the sensor network paradigm.

3.1 Open Shortest Path First

The OSPF protocol is based on a replicated link-state database model. Its complexity stems primarily from the database exchange and synchronization process. Peer sensors must synchronize their databases via link-state advertisements (LSA) before forwarding any application traffic. A link-state database forms a weighted graph of the network. Sensors perform Dijkstra's shortest path algorithm to build a routing table.

Neighbor discovery. Sensors broadcast hello packets at regular time intervals. A hello packet contains a list of known neighbors, i.e. nodes from which a sensor has previously received a packet, and targets all sensors within radio range. This way, sensors discover their neighbors automatically. There is no requirement for guaranteed delivery of hello packets. If the recipient sensor exists in the neighbor list, it asserts bidirectionality. A *hello interval* determines the frequency of hello packets. In the presence of other routing control traffic (e.g. a link-state advertisement) or application traffic, the hello protocol can suppress packets to avoid long message queues.

The OSPF protocol depends on soft state timers to detect a link or node failure. A neighbor entry is removed from the list, unless periodically refreshed. This timeout period is a multiple of the hello interval, e.g. 3 times the hello interval. Faster responsiveness to changes will require an explicit broadcast (e.g. a sensor may notify peers of its departure due to low energy levels), because if the timeout period equals the hello interval, the confidence of failure detection then relies heavily on lossless transmissions. Upon detection of neighbor failure, the protocol emits an LSA update to the network.

Database synchronization. If two sensors establish bidirectionality, they start a database exchange process, driven by a finite state machine (Figure 2(a)).

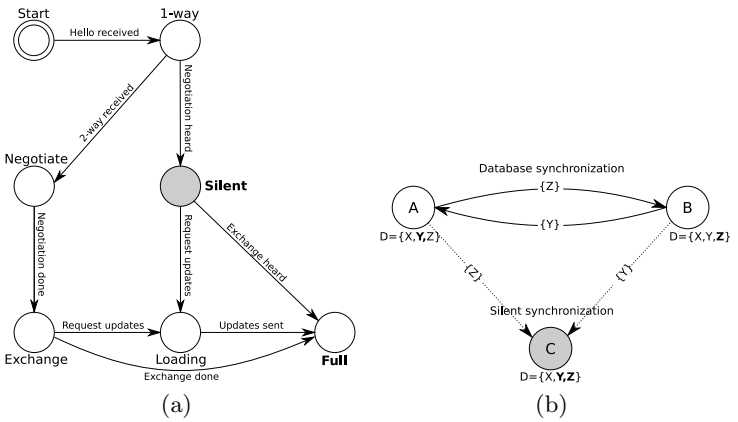


Fig. 2. The database exchange and synchronization process in the OSPF protocol.

Initially, sensors exchange database descriptions, rather than content, in a master/slave model. Only one outstanding description packet exists at a time. A slave sensor acknowledges the previous packet implicitly, by incrementing its master's sequence number in its next description packet. Eventually, every sensor receives a database summary of its peer. If there is a discrepancy, sensors load the missing or updated advertisements in update packets, in response to a request. Once they receive them, the two sensors become fully adjacent.

In a network of n sensors, the protocol must form $\frac{n(n-1)}{2}$ adjacencies. This causes a significant overhead to the sensing system. In a broadcast network, OSPF elects a designated router; all other routers synchronize with it, reducing the number of adjacencies to n. However, this would impose a hierarchy in our sensor network model. Sensors are capable of passive traffic monitoring within radio range, because they operate in a shared broadcast medium. In *silent synchronization*, a sensor can overhear a database exchange between two of its neighbors, viz. their description and update packets (Figure 2(b)). Thus, it becomes fully, or partially, adjacent. A partial adjacency is complete when the silent sensor sends its own updates, if any, to its neighbors.

Sensors flood update packets to the entire network to maintain network synchronization in its entirety. OSPF employs a simple, but reliable, flooding mechanism to disseminate state. Every sensor implicitly acknowledges receipt by reflooding the message. At long refresh intervals (i.e. days) sensor flush their entire database. Sensors can utilize neighbor state (i.e. the hello protocol) to further reduce the flooding overhead. In particular, a sensor s with a neighbor set N_s will suppress its transmission if, and only if, a neighbor $i \in N_s$ exists so that $N_s \subset N_i$.

3.2 Ad hoc On-demand Distance Vector

The AODV protocol reduces control traffic by originating path requests on demand. It does not broadcast packets blindly; instead, reply packets are unicast

during path establishment. Liveness monitoring of active neighbors uses stateless hello messages. Upon failure, it generates an error message to notify upstream sensors that use the broken path. Below, we discuss the path discovery, establishment, and monitoring process of the protocol.

Path discovery. An event or alarm will trigger a route request only if a valid path to the sink doesn't exist. Paths expire either implicitly, e.g. routing state timeouts, or explicitly, e.g. error packets sent. Sensors buffer their data packets until they receive a reply. There can be more that one outstanding request at a time, identified by a sequence number. Route requests are broadcast, constrained by a a TTL hop count limit to reduce flooding redundancy. By default, it is set to the maximum path length, or diameter, of the network. All intermediate relays compete with each other to become part of the best path; they increment an additive metric (e.g. hop count) and rebroadcast the request; meanwhile they cash the sender as the next hop to the originator. The AODV protocol uses implicit acknowledgements to determine bidirectionality between any pair of sensors. If a sensor marks, or blacklists, a link as unsteady, it ignores it in the path discovery process.

Path establishment. Upon receipt of a request, sinks establish a path in reverse: they propagate their reply using the previous sensor as the next relay to the originator. A sink receives more than one request per source. Usually, the first request traverses the fastest path. The sink can reinforce multiple paths to an originator. In turn, it can discard, use, or store these paths, based on the collection strategy. Reply messages are unicast, rather than multicast. Sensors can generate a gratuitous reply, if they already maintain an active path to the destination. This does not guarantee optimality, but improves responsiveness. Furthermore, sometimes is important to notify sinks of path interests to monitor abnormal traffic patterns.

Path monitoring. Active sensors, i.e. sensors that participate in an active path, assert connectivity by sending periodic hello messages. The rest of the network remains silent. If within a refresh time interval a sensor doesn't receive a routing state update, is assumes a failure and generates a route error packet. A failure can affect one or more data flows. Route error packets invalidate these affected paths in neighbor sensors. In turn, they can iterate the error message until all affected originators reissue a request. However, given sensor constraints, it is best to repair paths locally. Upon failure, sensors attempt to fix their affected paths by issuing a route request locally (Figure 3). Local repair concerns only sensors between a source and a failure. This means relay sensors will not issue a request back to the originator; they will rather correct their paths during the new path establishment process.

4 Identifying the cost factors

Specification of a sensor monitoring system is an iterative process between environmental scientists and system designers; it attempts to identify *what* is the desired functionality, and *how* to deliver it in order to provide service guarantees

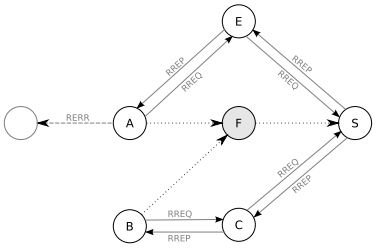


Fig. 3. Local repair in the AODV protocol. Upon failure of F, nodes A and B establish new shortest paths to sink S.

in a resource and time constrained environment. Functional requirements stem from the sensed area characteristics, the phenomena in question, or the data acquisition model – i.e. time-driven, query-driven, or event-driven model). The number of deployed sensors, the density of instrumentation, and the devices of choice form the energy, bandwidth, memory, and CPU constraints of the system in the aggregate. Non-functional requirements refer to end-to-end delivery guarantees, e.g. latency, or other application-critical considerations.

Functionally equivalent system components have different quality of service guarantees [14]. For example, on the selection of a routing protocol, reactive and proactive protocols exhibit similar functional behavior; they transfer messages from a source to a sink. But how can we make a choice? A vector-valued cost function describes the protocol behavior in the configuration space. Users can express their non-functional requirements as objectives or constraints over measurable system properties. This way, a solution is superior, or dominant, if it can better satisfy these objectives and constraints.[1]

The *hello interval* is a good candidate to drive the decision making process with regards to routing protocol choice as it affects both system objectives and constraints. In steady state, sensors exchange periodic messages to determine liveness of links. Recovery from link or sensor failures is directly related to the exchange rate of hello messages. It is clear that refresh timer intervals impact the overall routing protocol performance. In particular, high frequency trades off protocol responsiveness against additional energy and bandwidth wastage.[2] We further discuss the impact of the hello interval in Section 5.1.

Configuration settings (e.g. hello frequency) are hidden values during the specification process. They are the independent parameters in our evaluation strategy – in particular, we use the hello frequency and network size/topology. Initially, we consider two system properties, *time to repair* and *communication overhead*. Time to repair is the time to reach a steady state after a sensor failure; communication overhead is the number of message exchanges in total until

[1] The DIAS project (www.dcs.gla.ac.uk/dias) follows this design methodology of constructing heuristically optimal sensor systems with respect to a global cost function.

[2] Without loss of generality, we consider energy, whether stored or harvested, and bandwidth resources as system constraints, as they affect the overall system operational lifetime.

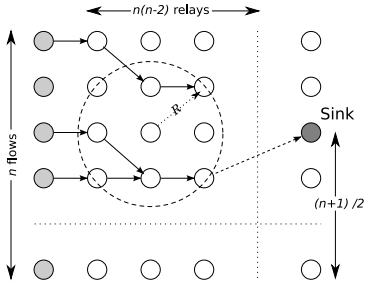


Fig. 4. A square grid topology. It consists of n^2 nodes, with $f = n$ sources transmitting readings to one sink, centered on the far edge, via $n(n-2)$ relays, with radio range R.

sensors establish an alternate path and is directly associated with energy and bandwidth resource consumption.

5 Simulation results

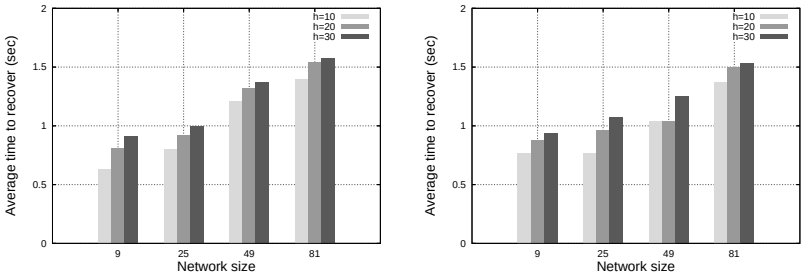
We conduct our study in TOSSIM [15], the TinyOS simulation environment. The network consists of n^2 sensors deployed in a square grid topology with a fixed internode distance. At every instance, we simulate n simultaneous flows, where n varies with the network size (Figure 4). The radio and channel parameters of the underlying link layer model mimic an outdoor application scenario (sensors in a football field) with low link asymmetry.

For the purpose of this study, we set the allowed hello packet loss to $d = 2$; it gives confidence regarding false positive (i.e. transient) errors, while it maintains fast responsiveness in our experiments. If a sensor does not receive an update within the interval $d \times h$, network state times out. We vary the hello message frequency h and present the average recovery time and average communication overhead from any relay failure, under different network sizes. $n(n-2)$ relays fail individually; this allows us to measure the overall recovery behavior of the model, in contrast to a random failure model. We simulate every failure point multiple times to gain confidence in our results.

If a sensor s fails at time t, the time to recover is $\Delta t = t' - t$. For OSPF, t' is the time until the last sensor, after notified about s's failure, recalculates its routing table. Note that OSPF calculates new paths to all sensors. For AODV, t' is the time until all s's neighbors re-establish their affected paths, if any. For example, if s does not participate in the routing graph, then $\Delta t = 0$. In AODV we simulate failures under the same routing graph. The packet overhead is the number of packets, inclusive of hello messages, sent during Δt.

5.1 The impact of hello frequency

Figure 5 shows the impact of the hello interval on the time to repair from an individual node failure for OSPF and AODV, respectively. The hello interval is set to $10s$, $20s$, and $30s$, and the network size to 9, 25, 49, and 81 nodes. Observe that low values of the hello interval improve protocol responsiveness to



(a) AODV (b) OSPF

Fig. 5. Normalized time to recovery for different hello intervals h.

failure. However, as we will see later, it adds control overhead. For lower values, i.e. $h < 10s$, hello packets dominate the network throughput, causing packets to drop, and consequently, unnecessary delays in the recovery process. For example, frequent hello packets can disrupt the synchronization or path discovery process.

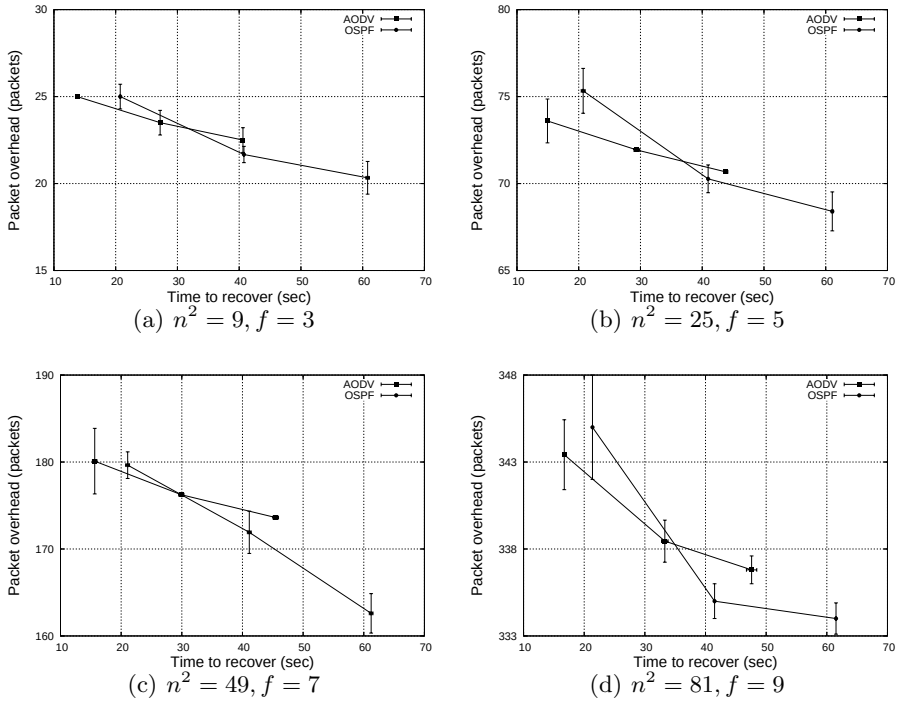
5.2 Making a selection

Although hello packets are small, they lead to congestion effects in the network: for small hello intervals, the protocol moves to higher packet loads. Figure 6 illustrates the trade off between time to recovery and packet overhead for varying hello intervals, viz. $10s$, $20s$, and $30s$. As an example, consider Figure 6(a), for $f = 3$. Assume that given an allowed time window for recovery (e.g. between two consecutive measurements), we wish to minimize the packet overhead. We can identify three regions of interest: the AODV area, for recovery time less than $\sim 22s$; the OSPF area, for recovery time longer than $\sim 42s$; and finally, an undecidable area, $22 - 42s$, where both protocols exhibit similar behavior.

Our experiments with the AODV protocol show that approximately 10% of relays remain idle; they are not active routers. Their failure will not affect network state and requires zero time to recover. Thus, the average time to recover drifts to lower values than the OSPF protocol. $\sim 10\%$ idle nodes also means that certain sensors relay one or more flows to a sink. The recovery time is now proportional to the number of affected flows. Simultaneous flows tend to use the same parts of a path. Figure 6 shows that AODV's time to recover increases with the number of simultaneous flows.

On the other hand, OSPF relays are active, even if they do not participate in an active path. So, they will all react to a failure by calculating new shortest paths. Furthermore, all n^2 sensors must periodically transmit hello packets, compared to AODV's selective hello protocol. For short intervals, hello packets occupy the wireless medium, causing fluctuations in the packet overhead of the recovery process (Figure 6). Its recovery performance is stable, bound by the $d \times h$ interval, and the number of sensors.

OSPF's regime of preference grows as the number of simultaneous flows increases (see Figures 6(a) and 6(d)). Graphically, as we move to higher loads, AODV shifts to the left, while OSPF remains stable. Also, the undecidable area



(a) $n^2 = 9, f = 3$ (b) $n^2 = 25, f = 5$

(c) $n^2 = 49, f = 7$ (d) $n^2 = 81, f = 9$

Fig. 6. Average time to recover vs. packet overhead for varying number of simultaneous flows f, and network size n^2.

shrinks, making a sharp contrast between the two protocols. Note that values in the recovery timeline correspond to an equivalent protocol configuration setting. In this example, time to recover maps to a specific hello interval. For the protocol of choice, we can extract appropriate configuration parameters to guarantee recovery timeliness. Given a particular configuration setting, the system must account for the expected packet overhead.

6 Discussion

In our simulation environment, sensors are arranged in the vertices of a uniform square grid area (Figure 4). Due to the regular spacing of sensors, this topology model minimizes interference amongst neighbors to the hidden and exposed terminal problems – an RTS/CTS MAC protocol can tackle these problems. However, an irregular physical layout can affect the performance of routing protocols. For example, random placement of sensors may result in longer hops with high bit error rate (BER) due to transmission power decay and interference; in this case, minimum hop count is not the best routing strategy – in particular, grids yield the best average BER performance, compared to random placement models [16]. Although a regular topology is fairly idealistic, it is not necessarily unrealistic. The spatial arrangement of environmental monitoring sensors is

usually deterministic. That is, dense clusters appear due to concentration of phenomena, not because of a random scatter of sensors.

Physical layer performance metrics, e.g. BER or packet loss, can change our routing semantics to avoid long, inefficient paths. These metrics are functions of the internode interference level. In our experiments, TOSSIM simulates a channel using a log-normal path loss model. Irregular placements favor proactive systems. For example, we might expect the number of simultaneous flows that utilize sensors close to a sink to be larger than the square root of the total number of nodes. This will cause the preference region for proactive protocols to further grow, at the expense of the reactive protocols.

Our implementations use soft state timers to dispatch packets from message queues to the network. By default, TOSSIM simulates a CSMA radio protocol; sensors compete to access the channel. Time synchronization causes the network to congest due to bursts of traffic (e.g. all sensors send hello messages simultaneously), resulting in packet collision and loss. To address this effect, we introduce a random clock drift among sensors. This time drift, similar to both protocols, is also projected in our measurements of recovery time.

The neighbor discovery and database exchange process of the OSPF protocol use both small and large packets, broadcasts and unicasts, to assert link quality. Their combination tends to detect link-layer problems more accurately, in contrast to simple keep-alive messages, by failing to form a full adjacency. This behavior makes transient connectivity problems easier to detect and, most importantly, path selection more robust. It also allows us to set the allowed hello packet loss to small numbers, thus improving the protocol responsiveness.

Let us consider the impact of management traffic in sensor routing protocols. It must be possible for the sensor system to transfer responsibility to an appropriate authorized system, if such control is asserted in the proximity of the network, to configure or re-program the system. If a management entity requires access to n nodes, from an arbitrary point at the network's boundary other than a sink, reactive protocols must establish n new paths. On the other hand, a proactive system is ready to address all sensors at any given time.

We further discuss support of proactive systems for management tasks in a scenario where a user adds a new sensor, or new relay, to fix a network deficiency, e.g. replace a faulty sensor. The new sensor first advertises its existence via a hello message; second, it synchronizes with a neighbor – at this point, the new sensor can communicate with any node; finally, its neighbor floods an update packet to notify the entire network of its new database entry. The time to add a new sensor or relay and the packet overhead of the process is constant.

Thus far, protocols form paths to a single aggregation point. Initial results show that in the AODV protocol, time to recover increases in the presence of multiple sinks. The OSPF protocol will not introduce any additional overhead; it supports many-to-many traffic by default. As the number of flows, or the number of sinks increases, protocols have the tendency to reuse the same paths, resulting in higher loads over a specific subset of relays. We wish to redistribute this load in support of the sensing application. Applications may also support

different classes of traffic with different delivery guarantees (e.g. delivery *vs.* delay). The OSPF protocol supports equal-cost multipaths by default; it maintains the necessary state to redistribute energy and bandwidth along different paths. If required, policies to exploit these equal-cost multipaths can be enabled.

7 Related work

Early attempts to apply OSPF to multi-hop wireless networks originate in the mobile *ad hoc* community. The Internet Engineering Task Force (IETF) proposed extensions to broadcast and point to multi-point interfaces to support mobility [17]. They use designated routers to reduce database exchanges and floods. The Optimized Link-State Retrieval (OLSR) protocol for ad hoc mobile sensor networks integrates the database exchange process to improve its reliability [18]; due to resource constraints, sensors exchange compact signatures of their database. Their work adds to OSPF for sensor networks: different representations of link state result in different memory and bandwidth usage.

The performance of the AODV protocol is tested in terms of packet delivery and energy efficiency in an indoor environment [19]. Results show that given sufficient routing state, AODV has acceptable packet delivery performance at the expense of energy wastage. The TinyADMR protocol [6] is another attempt to apply the reactive routing paradigm in sensor networks. In a similar manner, they implement an *ad hoc* protocol in TinyOS, taking into account the resource constraints of sensor networks.

Silent synchronization (Section 3.1) borrows from gossiping protocols for sensor networks [11, 9]. In contrast to probabilistic flooding, OSPF sensors are aware of their neighborhood; they can determine with confidence whether to broadcast or remain silent; whether to eavesdrop or actively exchange state. This work focuses on protocol behavior after they converge to a steady state; we do not present results on the initial overhead and delay until sensors reach this state.

The data aggregation paradigm improves energy efficiency in the presence of multiple flows to a sink. Data-centric routing protocols have been the subject of study in [8]. An aggregation strategy must be dictated by the application requirements. Routing protocols must support, rather than impose, different functional and non-functional services. With this policy-mechanism split in mind at design time, our protocols can be extended to support different types of service.

This work makes a sharp distinction between proactive and reactive protocols. Our taxonomy is orthogonal to existing routing paradigms. For example, the LEACH and TEEN protocols represent proactive and reactive routing in hierarchical distributed paradigms – hierarchical protocols delegate routing responsibility to a subset of sensors, the cluster leaders, based on proximity[20], or capability [5].

Multipath protocols are resilient to isolated or patterned failures because they can easily alternate routing paths [21]. They can employ either a proactive or reactive routing model. OSPF is able to discover multiple paths per sink, if they exist. However, the protocol does not impose their use. Also, path requests in the AODV protocol generate one or more paths to a sink. Given sufficient memory, sensors can maintain these multiple paths.

8 Conclusions & future work

There are regions in the time-to-repair $\times$ overhead space where the recovery process of the OSPF protocol performs better than the AODV protocol in the presence of multiple flows to a sink. In this region, the sensor system recovers from a failure faster and with less packet overhead, thus preserving scarce resources. To our knowledge, this is the first attempt to apply OSPF to sensor network routing. This proactive routing protocol for large-scale networks has proven a worthy adversary of its reactive counterparts for wireless sensor networks. This is primarily due to the manner in which proactive protocols disseminate state across the network, in contrast to reactive protocols who reveal state only upon request. Of course, sensors acquire network awareness at the expense of memory, energy, and bandwith resources; in return, the system has noticeable performance gains for its deployed lifetime.

Building an environmental sensor monitoring system, designers will always have a choice between proactive or reactive routing models. We construct a vector-valued cost function to automatically drive the selection process. Our framework selects a proactive or reactive protocol based on fault tolerance service goals. This approach is generic; in the future, we wish to further relax our configuration parameters and search for an appropriate protocol in this multi-objective problem space. Then, it may be necessary to seek Pareto dominant sets in this extended requirements space [14].

The initial design goal of OSPF was to build an extensible, descriptive routing protocol to support multiple types of service for large-scale autonomous networks. The dynamic nature of the protocol allows us to experiment with different metrics and service goals. We envisage a family of protocols to address two primary system requirements; networking support for query-driven application models, and co-design with low-level networking protocols. Support for query processing in sensor systems originates in careful exchange of sufficient metadata to optimally distribute packets among query injectors and projectors[22]. Moving down the protocol stack, topology control protocols can reduce the topology of a network, thus minimizing the control packet overhead. Furthermore, a routing protocol can synchronize with the ON-OFF periods of radio activity to conserve energy.

References

1. Culler, D., Estrin, D., Srivastava, M.: Overview of sensor networks. Computer **37**(8) (2004) 41–49
2. Martinez, K., Hart, J.K., Ong, R.: Environmental sensor networks. Computer **37**(8) (2004) 50–56
3. Heinzelman, W.R., Chandrakasan, A., Balakrishnan, H.: Energy-efficient communication protocol for wireless microsensor networks. In: Proc. of the Hawaii International Conference on System Sciences. Volume 8. (2000)
4. Manjeshwar, A., Agrawal, D.P.: TEEN: A routing protocol for enhanced efficiency in wireless sensor networks. In: Proc. of the International Parallel & Distributed Processing Symposium. (2001)

5. Younis, M., Youssef, M., Arisha, K.: Energy-aware routing in cluster-based sensor networks. In: Proc. of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems. (2002)
6. Chen, B., Muniswamy-Reddy, K.K., Welsh, M.: Ad-hoc multicast routing on resource-limited sensor nodes. In: Proc. of the International Workshop on Multi-hop ad hoc networks. (2006)
7. Hull, B., Jamieson, K., Balakrishnan, H.: Bandwidth management in wireless sensor networks. Technical Report MIT-LCS-TR-909, Massachusetts Institute of Technology, Laboratory for Computer Science (April 2003)
8. Krishnamachari, B., Estrin, D., Wicker, S.: Modelling data-centric routing in wireless sensor networks. In: Proc. of the IEEE Infocom. (2002)
9. Kulik, J., Heinzelman, W., Balakrishnan, H.: Negotiation-based protocols for disseminating information in wireless sensor networks. Wireless Networks **8**(2/3) (2002) 169–185
10. Intanagonwiwat, C., Govindan, R., Estrin, D., Heidemann, J., Silva, F.: Directed diffusion for wireless sensor networking. IEEE/ACM Transactions on Networking **11**(1) (2003) 2–16
11. Levis, P., Patel, N., Culler, D., Shenker, S.: Trickle: a self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In: Proc. of the Symposium on Networked Systems Design and Implementation. (2004)
12. Moy, J.: OSPF version 2 (April 1998) RFC 2328.
13. Perkins, C., Belding-Royer, E., Das, S.: Ad hoc on-demand distance vector (AODV) routing (July 2003) RFC 3561.
14. Ma, H., Wang, D., Bastani, F., Yen, I.L., Cooper, K.: A model and methodology for composition QoS analysis of embedded systems. In: Proc. of the IEEE Real Time on Embedded Technology and Applications Symposium. (2005)
15. Levis, P., Lee, N., Welsh, M., Culler, D.: TOSSIM: accurate and scalable simulation of entire TinyOS applications. In: Proc. of the International Conference on Embedded Networked Sensor Systems. (2003)
16. Panichpapiboon, S., Ferrari, G., Tonguz, O.: Impact of interference in uniform and random ad hoc wireless networks. In: Proc. of the IEEE Vehicular Technology Conference. (2004)
17. Spagnolo, P.A., Henderson, T.R.: Comparison of proposed OSPF MANET extensions. In: Proc. of the Military Communications Conference. (2006)
18. Clausen, T., Baccelli, E., Jacquet, P.: OSPF-style database exchange and reliable synchronization in the optimized link-state routing protocol. In: Proc. of the IEEE Conference on Sensor and Ad Hoc Communications and Networks. (2004)
19. Pham, N.N., Youn, J., Won, C.: A comparison of wireless sensor network routing protocols on an experimental testbed. In: Proc. of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing. (2006)
20. Mao, Y., Wang, F., Qiu, L., Lam, S.S., Smith, J.M.: S4: Small state and small stretch routing protocol for large wireless sensor networks. In: Proc. of the Symposium on Networked Systems Design and Implementation. (2007)
21. Ganesan, D., Govindan, R., Shenker, S., Estrin, D.: Highly-resilient, energy-efficient multipath routing in wireless sensor networks. Mobile Computing Communications Review **5**(4) (2001) 11–25
22. Woo, A., Madden, S., Govindan, R.: Networking support for query processing in sensor networks. Communications of ACM **47**(6) (2004) 47–52

Acknowledgments. The authors wish to acknowledge the support of the Engineering and Physical Sciences Research Council under grand EP/C014774/1.