Qualifying Dissertation: Knowledge Management in Wireless Sensor Networks

Jonathan Tate

June 2007

Contents

Abstract

Wireless Sensor Networks, or *sensornets*, are an emerging discipline of embedded system and network design. Large numbers of minimally resourced nodes are equipped with sensors to monitor their physical environment. Nodes cooperate to manage ad-hoc wireless networks, through which distributed applications distill voluminous raw data about sensed physical phenomena into meaningful information with utility to end-users. Some sensornet nodes are equipped with actuators with which to manipulate the physical environment, forming feedback loops under sensornet control.

This document examines literature published to date with regard to *knowledge management* in the sensornet domain. Current research challenges for viable sensornets include trading-off centralised and distributed data processing, resilience to network damage and adverse environmental conditions through highly adaptive designs, functioning with highly limited energy resources, recognising and extracting useful information among raw data, scaling to millions of nodes, working with data- and geography-centric paradigms, and ensuring that quality-of-service and timeliness concerns are adequately addressed. Proposals for future research are described, building on the substantial body of extant relevant work.

Qualifting Dissertation size restrictions necessitate relatively light coverage of interesting subjects in the interests of brevity. Readers seeking further detail are encouraged to consider the unexpurgated version of this document [235].

Chapter 1

Introduction

This chapter seeks to define the boundaries of the subject to be addressed in subsequent chapters of this Literature Survey, and hence the boundaries of the problem space to be addressed by the work proposed in the concluding chapter.

1.1 Wireless Sensor Network research

Research in *Wireless Sensor Networks*, or *sensornets*, necessarily touches upon many research topics of Computer Science and some of Electronic Engineering, drawing upon the existing work in related fields. However, the unique characteristic of the Wireless Sensor Network field is the interplay and integration of these foundation subjects, yielding a distinct topic worthy of further study in its own right. From the amalgamation of existing work in such diverse subjects as *computer networks*, *energy management*, *wireless communications*, *distributed computation*, *data management*, and numerous others, emerges the topic of sensornet research with its own goals, challenges, and opportunities, distinct from those of the related work from which it is drawn. Sensornets are typically *highly scalable*, *energy-aware*, *geography-aware*, *data-centric*, *application-specific*, *self-configuring*, and *self-adapting.*

1.2 Terminology

Tilak et al [238] provide a taxonomy of *sensornet* models, in which they define the following terms:

- **Sensor** - an independently-operating device that implements the physical sensing of environmental phenomena and reports measurements through wireless communication. Typically consists of five components: sensing hardware, memory, battery, embedded processor, radio transceiver

- **Observer** - the end user interested in obtaining information disseminated by the sensor network about *phenomena*, phrased as *interests* to the network
- **Interests** - a query regarding physical phenomena sensed by the network, addressed to the sensornet as a single entity rather than to a specific *sensor*
- **Phenomenon** - the entity or issue of interest to the *observer* that is sensed and potentially analysed/filtered by the sensornet
- **Measurements** - a set of discrete sample values taken from a single *sensor*
- **Source** - the start-point of a communication within a sensornet, typically a producer of *measurement* data or *interests* data
- **Sink** - the end-point of a communication within a sensornet, typically a consumer of *measurement* data or query response data

1.3 Characteristics

Wong and Arvind [120] state that the defining characteristics of a *sensornet* are:

- **Data-centric**: The primary focus of a sensornet is to produce data about the physical environment in which the network resides by taking readings from sensors attached to network nodes. Often the processing of this data will occur at a central hub, although in-network processing is possible.
- **Inter-node communication**: Nodes can communicate with their neighbours within a physical distance limited by wireless communication methods, typically of the order of metres but almost always less than the physical diameter of the network (hence ruling out a point-to-point network model)
- **Mobility**: Sensornet nodes are largely static, although some nodes may be mobile, and hence the sensornet is largely a static network notwithstanding node failure or temporary disconnection for power conservation.
- **Data transfer**: Sensor data is produced at *source* nodes and is transported to *sink* nodes at which it is consumed or otherwise processed. Often the *sink* node is a central hub, although a fully decentralised peer-to-peer model is feasible.

Estrin et al [69] state that a *sensornet* is also **application specific**. Rather than accommodating a wide variety of applications, a sensornet is typically designed and deployed with a specific task (or set of tasks) in mind. This allows the designer to optimise the network design to these specific goals, and implement application-specific in-network processing distributed across the nodes of the network. This is in contrast to the application-agnostic routers of a traditional data network, which merely forward data unchanged and hence cannot implement application-specific optimisations. In sensornets, data transmission is typically more expensive than data processing [186, 155].

Kahn et al [121] also note that sensornet nodes must *consume extremely low power*, *operate in high volumetric densities*, *have low production cost and be disposable*, *be autonomous and operate unattended*, and *be adaptive to the environment.* These requirements influence sensornet hardware and software design such that both should be considered simultaneously. The extreme nature of the problem, extreme scale of deployment, and extreme resource constraints, have required researchers to seek new solutions rather than recycle existing solutions from related problem domains.

1.4 Applications

Sensornet applications typically involve the distribution of a number of sensor nodes in a geographic region, where each node gathers raw data about its surroundings. Nodes may be equipped with seismic, magnetic, thermal, visual, infrared, acoustic, radar or GPS sensors [7]. These raw data can be processed and interpreted in-network to monitor a wide range of ambient conditions including temperature, humidity, vehicular movement, lighting condition, pressure, soil makeup, noise levels, presence or absence of certain types of objects, mechanical stress levels, speed, direction, or object size [69].

Akylidiz et al [7] examined the published literature and reported the following list of proposed applications:

- **Military applications**: Sensornets can be an integral part of C4ISRT systems (command, control, communications, computing, intelligence, surveillance, reconnaissance and targeting), performing tasks such as *monitoring friendly forces*, *battlefield surveillance*, *reconnaissance of opposing forces and terrain*, *targeting*, *battle damage assessment*, *nuclear/biological/chemical attack detection.*
- **Environmental applications**: *Forest fire detection*, *environment biocomplexity mapping*, *flood detection*, *precision agriculture.*
- **Healthcare applications**: *Telemonitoring of human physiological data*, *tracking patients and doctors*, *drug administration in hospitals.*
- **Home applications**: *Smart environment*
- **Commercial applications**: *Environmental control in offices*, *detecting and monitoring car theft*, *managing inventory control*, *vehicle tracking and detection.*

Culler and Estrin [54] provide a classification of sensornet applications which is more generic, albeit rather more simplistic. They consider applications to be differentiated into the categories:

- **Monitoring space** e.g. *environmental monitoring*, *habitat monitoring*, *precision agriculture*, *indoor climate control*, *surveillance*, *treaty verification*, *intelligent alarms*

- **Monitoring things** e.g. *structural monitoring*, *ecophysiology*, *condition-based equipment maintenance*, *medical diagnostics*, *urban terrain mapping*
- **Monitoring the interactions of things with each other and the encompassing space** e.g. *monitoring wildlife habitats*, *disaster management*, *emergency response*, *ubiquitous computing environments*, *asset tracking*, *healthcare*, *manufacturing process flow*.

From the above, we see that even a high-level of classification of sensornet applications can encompass multiple dimensions, as we categorise a set of candidate applications against sets of assessment criteria. Care must be taken in assessing the results obtained in sensornet research to separate the application-dependent and -independent factors, and to avoid confounding of sensornet factors among environmental factors.

Many other proposals for applications of sensornets have been published, in numbers too great to list comprehensively. However, some of the more interesting work describes real deployments of sensor networks. Although of limited scale, they provide interesting insight into the strengths and weaknesses of the sensornet concept and the current capability of realising these networks in practice.

Habitat monitoring sensornets [39] were deployed to monitor the ecology of the Leach's Storm Petrel [232, 231]. Interestingly, the investigators observed that the node failure rate was much higher than expected, and that unexpected emergent behaviour was observed which could not be predicted without *building complete systems and deploying them in realistic conditions* [231].

Wark et al [245] describes an interesting Wireless Sensor-Actuator Network deployment, monitoring and influencing livestock behaviour through per-animal actuators. Closing the sensor-actuator loop in this manner provides an interesting challenge for the modelling and simulation of the network. Accurately predicting livestock reactions, even within short periods, requires far greater modelling detail than applied in purely observational longer-term studies [232, 231]

Other proposed sensornet applications include the *Smart Kindergarten* project [25] to provide a rich problem-centric learning environment, monitoring underground structures for failure [140], detecting and tracking plumes of gaseous substances in the environment [216], detecting ruptures in pipelines [226], structural health monitoring [70, 123, 169], automated distributed surveillance [106], biomotion capture [13], tracking moving objects [215, 132, 67], and lighting control for media production [177]. This is not an exhaustive list, and countless other applications could be conceived.

1.5 Related work

Sensornets are similar to *Mobile Ad-hoc Networks* (MANETs) in that they are composed of small, low-resource computer nodes connected through a wireless network. Akyildiz et al [7] consider MANETs to be *the closest peers to sensor networks*, noting that MANETs place far greater importance on mobility but may be composed of far fewer nodes, such that *none of the existing Bluetooth or MANET MAC protocols can be directly used* in a sensornet.

Smart Dust is an emerging technology of wireless sensing networks, similar to a conventional *sensornet* [246]. They utilise nodes that are physically very small, on the order of millimetres, and are deployed in a correspondingly small physical area, typically of the order of metres [121] rather than the kilometres of *sensornets* [120]. Other than size, perhaps the most significant difference between a *Smart Dust network* and a conventional *sensornet* is the medium for wireless communications; *Smart Dust* nodes typically communicate by optical transmission rather than the radio-frequency communication of *sensornets* [121].

Specknets, or *Speckled Computing Networks*, are a dense and non-static wireless network of thousands of very small, resource-constrained nodes called *specks* collaborating to process data [120]. *Specknets* resemble *sensornets*, but are *application-centric* rather than *data-centric*, have shorter internode communication range (tens of centimetres), are highly mobile, and apply a fully decentralised peer-to-peer model dissimilar to the *sources* and *sinks* of sensornets. However, it could be argued that *specknets* are merely a subset of *sensornets* rather than a separate discipline.

Active networks are computer networks in which *the switches of the network perform customized computations on the messages flowing through them* [236]. Such networks are considered *active* because *nodes can perform computation on, and modify, the packet contents*, and are considered by Tennenhouse et al [236] to be *a natural step beyond traditional circuit and packet switching.* It is evident that sensornets are also *active networks* if any in-network filtering, aggregation or other processing is implemented.

Distributed computing divides a computation task into smaller subtasks, allocates these subtasks to independent computing resources, then compiles the set of subtask results into a result for the initial task. Any sensornet which performs in-network aggregation or collaborative sensing could be considered a *distributed computing system*, under Stankovic's definition of *a collection of computers connected by a communications subnet and logically integrated in varying degrees by a distributed operating system and/or distributed database system* [223]. However, this definition implies a somewhat more thorough level of integration and coordination between computers than is necessarily evident in the highly adaptive environment of a sensornet, and does not require that the computers sense or influence their physical environment.

Distributed databases and *parallel databases* are similar to conventional databases in that they manage, archive and query data, but distribute the data and processing across multiple physical nodes [175]. The motivation for distribution varies, but is typically to make the dataset available at multiple physical locations, to enable processing of a dataset which is too large for any single node, to provide resilience to equipment failure through redundancy, or to enable load-balancing. Sensornets do not necessarily implement *distributed database* functionality, but some applications will require this. Barbará [19] considers that the main challenges in the wireless mobile context are *communication asymmetry*, *frequent disconnections* and *power limitations.*

Real-time databases are databases which exhibit predictable behaviour, conform to explicit timing constraints, and apply time-cognisant protocols to handle *deadlines* and *periodicity* *constraints* associated with database activity [191]. A *temporal database* contains *time-varying data* [174] which has *temporal validity* [191]. Ramamritham [191] suggests that the time-dependent characteristics of *real-time* and *temporal* database behaviours are strongly linked, as most *real-time databases* are inherently *temporal databases*, whereas Özsoyoĝlu and Snodgrass [174] perceive these to be distinct research topics that may be applicable to a given system but are otherwise separate. Necessary interaction between sensornets and their physical environments induce real-time requirements, hence many sensornet applications must maintain *real-time databases.*

Both *real-time* and *temporal* databases extend conventional database theory in the time domain, with *temporal database theory* considering the impact of time upon data, and *real-time database theory* considering the impact of time upon the database system which manages data. Although there is generally no requirement that the data managed by a *real-time database* be *temporal data*, this pairing is likely in sensornets producing and processing data for real-time consumption or controlling actuators. In addition to the general real-time scheduling theory reapplied to database tasks, *real-time databases* and *temporal databases* must consider the notion of *temporal consistency* [191].

Safety critical system design techniques seek to quantify, control, and minimise, the possibility of system failure. This is generally undertaken where the failure or malfunction of a system could cause or allow unacceptable risk or damage to people or other entities, and expose the system operator to financial or legal penalties [25]. *Sensornets* yield data about the real world which, if incorrect or incomplete, could lead the network operator to incorrect or inappropriate conclusions. Of more concern would be *Wireless Sensor-Actuator Networks* which interact with and modify their environment, and for which special consideration must be given to the potentially deleterious effects of system misbehaviour.

1.6 Document structure

As noted above, research in sensornets draws upon the considerable body of work extant in related and diverse research disciplines. Whereas understanding of sensornets is predicated on an awareness of this related work, it is self-evident that it is beyond the scope of this document to describe fully both the breadth and depth of this body of important work. Instead, the remainder of this document selects a small subset of topics of particular significance to sensornets, describing in particular the influence of these fields as they relate specifically to sensornets.

Other topics not selected for inclusion in the latter chapters of this document retain importance and influence on sensornet design, but are omitted to retain sharp focus on knowledge management within sensornets. Other research topics bear upon sensornet design either directly or tangentially, and their influence is implicit where not specifically addressed.

Chapter 2, *Data aggregation*, addresses in-network processing strategies for aggregating source data sets into smaller result sets, where the latter retain the key characteristics and information of the original data. In-network data aggregation strategies must consider both aggregation methods and their application in distributed processing environments.

Chapter 3, *Data-centric routing*, examines content-aware data routing methods. The content of data traversing the network influences routing decisions, and must address the needs of both data producers and data consumers.

Chapter 4, *Peer-to-Peer networks*, considers peer-to-peer data networks constructed for sensornet applications. Peer-to-Peer networks constructed using Distributed Hash Tables enable efficient data-centric mechanisms and implicitly encourage replication for enhanced performance and reliability, but typically offer poor support for approximate queries characteristic of sensornet applications.

Chapter 5, *Scheduling and real-time guarantees*, examines the need for real-time guarantees and Quality of Service contracts in sensornet applications, with particular reference to sensornet characteristics which make traditional real-time and scheduling approaches difficult.

Chapter 6, *Cross-layer challenges*, discusses design issues and challenges which cut across all layers of sensornet designs, regardless of the selected design model or system modularisation. Successfully addressing cross-layer challenges may require holistic approaches with influence permeating throughout all sensornet elements operating in concert.

Chapter 7, *Conclusions and future work*, draws upon issues raised by the discussion of chapters 1-7 and outlines the direction of future work to build upon this knowledge. Opportunities for further work are discussed which would, if implemented, extend the findings of previously published work and research themes. Specific examples of possible work for implementation in the short term are provided, with discussion of broader themes for a longer-term outlook.

Chapter 2

Data aggregation

Sensornets may generate unbounded quantities of data, deriving from direct observation of the physical environment. Generally, sensornet users are not interested in raw sensor data; it must be processed into useful information. Implementing this processing within the network itself has the desirable outcomes of distributing workload and reducing the volume of data traversing the network, conserving precious resources. *Data aggregation* is *the combination of data from different sources* [14].

2.1 Aggregation functions

Aggregation functions are composed of merging function, f, initialiser i, and evaluator e [154]. Any operation expressible as commutative applications of a binary function can be expressed as:

$$\langle z \rangle = f(\langle x \rangle, \langle y \rangle)$$

where $\langle x \rangle$ and $\langle y \rangle$ are multi-valued *partial state records*, computed over one or more sensor values, representing intermediate states over values required to compute aggregate $\langle z \rangle$ [154]. Initialiser i specifies state record initialisation for a single unaggregated source value, and evaluator e takes a partial state record and computes the actual aggregate.

Madden et al [154] categorise aggregates against properties of *duplicate sensitivity*, *tolerance of loss*, *monotonicity*, and *state requirement*:

1. *Duplicate insensitive* aggregates are unaffected by duplicate readings, whereas *duplicate sensitive* aggregates change if duplicate readings are reported. This property influences need of networks guaranteeing *exactly-once* message deliver, and restricts possible optimisations.

2. *Exemplary* aggregates return one or more representative values from the set of all values, whereas *summary* aggregates compute some property over all values. *Exemplary* aggregates behave unpredictably in the face of loss and are not amenable to sampling, whereas *summary* aggregates of a subset of data are robust approximations of the aggregate of entire data sets. Sensornet designers may favour *exemplary* aggregates operating on small data subsets to reduce energy cost, or processing streamed data.

3. *Monotonic* aggregates have the property that when two partial state records s_1 and s_2 are merged by f the resulting state record $s\prime$ has the property $\forall s_1, s_2, e(s\prime) \geq MAX(e(s_1), e(s_2))$ or $\forall s_1, s_2, e(s\prime) \leq MIN(e(s_1), e(s_2))$. This determines whether some SQL-style predicates (e.g. *HAVING*) are applicable *in-network* before final aggregate value is known. Early predicate evaluation reduces messages by reducing distance traversed by partial state records in aggregation trees.

4. *State required for each partial state record* bears upon required storage and network resources, with performance typically inversely proportional to intermediate state required per aggregate. Five aggregate categories with regard to state:

   (a) *Distributive* aggregate partial state record size is equal to final aggregate size; partial state is simply the aggregate for the partition of data over which the aggregate has been computed.

   (b) *Algebraic* aggregate partial state records are of constant size; partial state records are not themselves aggregates for the partitions.

   (c) *Holistic* aggregate partial state record size is proportional to the size of data in the partition. For *holistic* aggregates no useful partial aggregation is possible, so all raw data must be collated at a single point.

   (d) *Unique* aggregates are similar to *holistic* aggregates excepting that propagated state size is proportional to the number of *distinct* values in the partition.

   (e) *Context sensitive* aggregate partial state record size is proportional to some property, perhaps statistical, of data values in the partition. Many *approximate aggregates* are *context-sensitive.*

Any aggregation function, including common SQL aggregation functions including *MAX, MIN, COUNT, SUM, AVERAGE, MEDIAN,* and *COUNT DISTINCT*, can be positioned in this four-dimensional space [154] which for obvious reasons is not illustrated here.

Culler et al [156] consider challenges of implementing five basic database aggregates of *MAX, MIN, COUNT, SUM, AVERAGE*, and *COUNT* with *grouping* in sensornets. These aggregates are well-suited to sensornets, being expressible as aggregate functions, f, mapping readily to network implementations where

$$f(a \cup b) = g(f(a), f(b)) \tag{2.1}$$

Naive aggregation implementations in sensornets would forward all source data to base stations for centralised, server-based processing [156]. Instead, Heidemann [102] proposes partially or fully computing aggregates as data traverses the network to reduce latency and energy cost. Culler et al believe *in-network aggregation is always a superior choice* [156]. An interesting question is whether similar benefits are observed for more general in-network processing structured similarly. *In-network aggregation* consists of two phases; a *propagation phase* where *aggregate queries* are pushed down into the sensornet, and an *aggregation phase* in which *aggregate results* propagate upward from children to parents. *In-network aggregation latency* may be *several seconds* for sensornets. Multiple *aggregation phases* may result from a single *propagation phase*, reducing energy costs and latency.

Sensornets are *inherently unreliable* [156], rendering impossible attempts to ensure that any given subnetwork contributes to a given aggregate. Reattempting failed aggregation within subnetworks by *multiple aggregation* may address transient failures, but increases energy cost and latency by factor x for x reattempts, and cannot address nontransient failures.

An alternative remedy is *pipelined aggregation* with modified *aggregation phase* [156]. Time is divided into intervals of duration i. During each interval, each sensor which *at this time* received the query during *propagation* immediately returns the partial aggregate from its own readings, and values provided by its children in the aggregation tree. *Propagation* and *aggregation* phases now overlap. Each interval, the base station receives a new aggregate value of monotonically increasing accuracy, calculated from a greater proportion of the network. A useful approximate aggregate is available early, latency is reduced, with lower energy cost than *multiple aggregation* but slightly greater than *simple aggregation.* Culler et al [156] do not consider aggregating *fully pipelined communications* of streamed or incomplete data, or sending updates only *only when sensor readings change* significantly. Event-detection sensornets would benefit greatly from the latter, preventing unnecessary aggregate refinement when existing aggregates are of sufficient accuracy for event detection.

2.2 Data aggregation strategies

Data aggregation strategy metrics include *energy savings*, *delay*, and *robustness.* Introducing data aggregation strategies into system designs fundamentally changes system behaviour [167]. If data aggregation is required, it should be considered throughout system design. Data aggregation results in fewer transmissions, but can increase latency if data from nearer sources are held back at an intermediate node in order to be aggregated with data from more distant sources [14].

An *optimal aggregation strategy* [14] employs aggregation functions transforming one or more incoming data packets into exactly one outgoing data packet. Each node in the data aggregation tree, formed between multiple sources and the single sink, transmits exactly once; the tree is essentially the reverse of a multicast tree. The optimum number of transmissions per source datum is edge count for the minimum Steiner tree containing the source and sink nodes. Unfortunately, finding this minimum Steiner tree is NP-hard for general network graphs with arbitrary node placement, but comparison of suboptimal schemes against this lower bound are possible. Suboptimal polynomial-time schemes include:

- *Center at Nearest Source (CNS)*: Source nearest the sink is aggregation point; all other sources send directly to this aggregation point.
- *Shortest Paths Tree (SPT)*: Each source sends data to sink along shortest path between the two. Where paths from distinct sources overlap, a vertex is formed in the aggregation tree.
- *Greedy Incremental Tree (GIT)*: Aggregation tree is built sequentially. In the first step, the tree consists of the shortest path between sink and nearest source. At each subsequent step, the next nearest source closest to the current tree is connected.

Where sensornets are organised into clustered topologies, clusterhead nodes can collect and aggregate values contributed by other cluster nodes for onward processing. Mharte and Rosenberg [167] define function $\chi(x)$ to calculate total packet count during each cluster data gathering cycles, where outgoing packet count is proportional to input packet count as *compression ratio* m, and c is constant overhead per aggregation cycle:

$\chi(x) = mx + c$

Aggregation performance depends mainly on m:

- $m = 0$: Any number of packets are condensed into a single packet. Example conformant aggregating functions include *min*, *max*, *sum*, *mean*, or boolean decisions.
- $m < 1$: Compression ratio is fixed, reducing packet transmission by factor m.
- $m = 1$: Packet count unchanged by aggregation function. Mhatre and Rosenberg opine this implies *no aggregation*, but this untrue; clusterheads would block forwarding of source packets until all cluster source packets were available, perhaps increasing latency but requiring fewer energy-hungry radio state changes.
- $m > 1$: Mharte and Rosenberg discount this possibility, with implied data size increase. Although perhaps atypical, this possibility is nevertheless feasible. For example, clusterheads could annotate original source data with aggregate summaries.

This linear model is overly simplistic, discounting variable compression ratios, aggregation functions yielding multiple output packets, and invariance of packet size. More fundamentally this model considers only single clusters, whereas system designers must consider entire aggregation trees in hierarchically-clustered networks.

Intanagonwiwat et al [116] consider the relative merits of *opportunistic aggregation* and *greedy aggregation*. Under *opportunistic aggregation*, low-latency paths between *sources* and a *sink* form an *aggregation tree* rooted at the *sink*. Whenever similar data meet at branching points, multiple instances are replaced by single aggregate instances. However, efficiency is suboptimal as similar data may only cross high in aggregation trees.

Instead, *greedy aggregation* [116] constructs *greedy incremental trees* where only the first shortest source-sink path is constructed. Subsequent sources incrementally connect to the existing tree using *directed diffusion* approaches. An *incremental energy-cost field* is maintained allowing joining sources to identify the nearest extant tree vertex. *Path pruning* removes redundant paths by *negatively reinforcing* inefficient or inactive neighbours. *Greedy aggregation* yields similar *delay* and *distinct-event delivery ratio* as *opportunistic aggregation*, but can reduce energy cost by 45%.

2.3 Overcoming difficulties in aggregation

Common aggregation functions are vulnerable to attacks which adversaries insert erroneous values, hide real values, or otherwise influence aggregate values. Wagner [244] examines standard SQL aggregation functions, finding *COUNT* secure against source data modification, and *MAX*, *MIN*, *SUM*, and *AVERAGE* vulnerable. However, it is difficult to support Wagner's assertion of *COUNT* security for in-sensornet aggregation. Without unique node identification, any given node may deliver any number of indistinguishable datum-bearing messages, perhaps sending none due to random failure or energy management policy. Aggregation root nodes cannot possibly determine how many unique messages to *expect*, only count those actually *received* within deadlines.

Wagner [244] states *aggregation primitives [should] be viewed as nothing more than statistical estimators.* Applying *robust statistics* principles, a *resilient aggregation* approach can cope with *noisy and error-prone data.*

Under Wagner's proposals [244], an estimate can be placed on the root-mean-square error introduced by a given aggregation function if certain properties are known of the network, as the extent to which a single incorrect value can influence the overall aggregate is determined mainly by the selected aggregation function. The aggregation function should be selected to minimise the anticipated error. Wagner suggests that a *truncated aggregate* can be found by rejecting any data values falling beyond acceptable defined upper and lower bounds. However, this approach assumes that these bounds can be known in advance, and does not prevent an attacker skewing the aggregate observation within these bounds. Instead, a *trimmed aggregate* can be obtained by rejecting the minimum and maximum $n\%$ of raw observations prior to applying any aggregating function. This has the advantage that the network operator does not need to define upper and lower acceptable bounds, and would require many nodes to malfunction or be compromised by an attacker to influence the aggregate value, but would require each aggregating node to buffer values and hence increase storage overhead and latency.

However, in both *truncated* and *trimmed* approaches there is an inherent assumption that unusual values are outliers that can be safely discarded. If a sensornet is deployed to observe for the occurrence of unusual events then these approaches will discard the most valuable raw data. A related assumption is that the distribution of raw values will be centred on some intermediate value and tail off on either side. Unless these assumptions hold true in the actual deployment environment, the derived aggregates will be misleading.

Hu and Evans [111] recognise that in-network aggregation requires each node to have access to all information, so it is not possible to secure aggregation by using end-to-end encryption with a unique key for each leaf node. An important consequence is that each node must contain authentication keys applicable across the network, so an attacker with modest technical capability could compromise a single node, extract the keys, and use these keys to insert erroneous values into the network. Hu and Evans propose to address this weakness by using *delayed aggregation* and *delayed authentication*.

Under *delayed aggregation*, rather than aggregating data at each hop, data are forwarded unchanged over a first hop and aggregated at the next hop [111]. The original source node can detect if the next-hop node forwards the data unchanged. Under *delayed authentication*, messages are not authenticated immediately; the authentication keys are only transmitted after some delay, if no reports of suspicious behaviour reach the base station within this delay. This increases latency, but is intended to limit the possible damage from compromise of a single node by allowing each node to have a unique key. Of course, broadcasting the authentication keys at this later time negates much of the benefit of delaying authentication if the keys are reused for subsequent communications.

Hu and Evans' approaches [111] increase the transmission costs of data in the network, increase the per-node storage overhead, and do not provide any safeguards against incorrect data from malfunctioning nodes. Furthermore, it is assumed that a base station can transmit with sufficient radio power to reach all nodes simultaneously, and it is difficult to accept this assumption in the case of a general sensornet. However, *delayed aggregation* and *delayed authentication* do guarantee integrity where network routes will never pass through two consecutive compromised nodes. However, in a sensornet it is reasonable to assume that if an attacker can compromise a single node then they may also be able to compromise further nodes nearby.

Yang et al [253] take a slightly different approach in the *Secure hop-by-hop Data Aggregation Protocol* (SDAP). They recognise that low-latency and low-traffic aggregation requires per-hop aggregation, but this risks *false data injection* attacks in which incorrect values are inserted and *event suppression attacks* in which data corresponding to observed events are discarded or modified to mask the event. Interestingly, they observe that a compromised sensor node behaves in a similar manner to a faulty sensor node, suggesting that a single mechanism could address both concerns simultaneously.

The mechanism proposed by Yang et al [253] requires the aggregation tree to be divided at random into subtrees, forming logical groups in which aggregation occurs. The identity and membership of these groups is changed periodically, each time selecting a random node to act as group leader, minimising the damage that can be done by a single compromised node. Yang et al allow aggregation to occur unhindered within these groups, but then apply controls to aggregation *between* groups. If the sub-aggregate produced by a group is considered *suspicious*, then the entire group sub-aggregate is discarded. This is similar to the approach proposed by

Wagner [244] but applied in-network, and operating at a higher level of network structure than the individual node.

However, this approach has some weaknesses. Discarding all values from a group of nodes, rather than a single node, risks discarding all observed data (including genuine data) from significant portions of the network. If the aggregation tree is composed of per-hop links that reflect the underlying geographical structure of the network, there will be holes in the geographic coverage of the aggregate. Yang et al's method [244] also suffers from the weakness that it must be possible in some way to decide that a sub-aggregate value is *suspicious*. Yang et al recognise that unusual values may be the most valuable, perhaps corresponding to observations of interesting events in the physical world, and yet their detection method is in essence an enhanced form of outlier detection. As such, it is difficult to support their claim of *zero false positive* detection.

Deligiannkis and Kotidis [60] indicate that *in-network aggregation* is most suitable for *exploratory, continuous queries that need to obtain a live estimate of some (aggregated) quantity.* They propose a simple aggregation framework, of which the most notable feature is support for estimating deviation of the aggregated signal from the actual measured signal. A tree of nodes is formed below a parent node, in which each subtree is responsible for aggregating the values obtained from each of its subtrees, down to leaf nodes which collect raw sensor data. Each node N_i transmits its partial aggregate to its parent node in the aggregation tree if and only if the subtree value deviates by a value greater than the permitted maximum error for this node, supressing network traffic until *interesting* readings of the partial aggregate are observed.

An *error budget* is allocated within the tree, assigning more permitted error to the nodes *expected to reduce their bandwidth most* through this process. Nodes with particularly large variance are excluded completely is reportedly essential for this algorithm to function, although this does raise the concern that it may exclude the most *interesting* and potentially useful data, in addition to excluding potentially malfunctioning nodes. However, this approach will fail if it is impossible to build the aggregation tree without exceeding the *error budget*, and offers no mechanism to resolve this conflict.

2.4 Selection of aggregator nodes

Chen et al [252] outline a framework for in-network data aggregation, in which the aggregation processing activity is delegated to a subset of the sensor node population dedicated entirely to aggregating data produced by other nodes. Their goal is to minimise the overall energy cost of transporting information throughout the network. Aggregation is intended to remove redundancies in data that result from spatio-temporal correlations, thereby reducing the volume of data to be transported in the network but retaining the essence of the original source data in the smaller aggregated product.

Chen et al define three types of data collection in sensor networks [252]:

- *Event-based data* - an observation of a specific event at a specific location and time
- *Focused state-based data* - data collected in response to a query directed to a specific set of nodes
- *Global state-based data* - data collected by sensors at all nodes within the deployment area

Their aggregation technique is of relevance only to *global state-based data*, and is therefore unsuitable for sensornet applications in which *event-based data* or *focused state-based data* predominate.

The first variant assumes a relatively flat network hierarchy in which nodes are organised into *clusters*, and the *cluster head* node is responsible for aggregating data produced by the cluster, and forwarding the compressed result onward to the data sink. In the second variant, the aggregators are organised in a hierarchy. The lowest level of aggregators are *cluster head* nodes as in the first variant, but a subset of these *level 1* aggregators are elected as *level 2* aggregators to aggregate the output of the *level 1* aggregators. This pattern continues at increasingly high levels of aggregation up to the sink node.

The algorithm for selecting aggregators at any level of the hierarchy of either variant is a randomised and fully distributed algorithm, operating in two phases. In the first phase, each node chooses to be an aggregator with probability p_1 independently for $p_1 \in [0, \frac{k}{n}]$, where k is the number of aggregators at this level of aggregation, and n is the number of nodes in the network. In the second phase, each node that finds itself not within the coverage radius of an aggregator declares itself as an aggregator with probability p_2. Chen et al [252] state *by careful choice of* p_1 *and* p_2*, we can ensure that the expected number of aggregators is* k*.*

k, p_1 and p_2 must all be calculated offline prior to network deployment, and are dependent on *number of sensors, deployment area, compression ratio, characteristic distance, and other network parameters.* This appears to suggest a major weakness in the approach of Chen et al. Firstly, this information may not be available a priori prior to deployment. Secondly, Chen et al do not provide a mechanism through which the selected values of k, p_1 and p_2 can be calculated in an application-independent manner. Thirdly, this information may change during the lifetime of the network, noting that the network application may also change over time or in response to observed activity.

Perhaps most significantly, this approach does not take account of node failures. The value calculated offline are partially derived from node count, but node failures or power management policies in the field will change the number of nodes active at a given time, invalidating the values calculated offline prior to deployment. Also significant is the disproportionate impact of certain nodes failing. If we assume that all nodes are equally likely to fail, we find that the failure of an aggregator has more impact than the failure of a non-aggregator, as the algorithm does not allow for failure detection or clusterhead re-election. This is critically important in the hierarchical case, in which failure of a single node acting as a high-level aggregator will effectively silence the data from all sub-aggregators and non-aggregators below it in the aggregation hierarchy.

Another significant issue is that of uneven energy consumption. Chen at al [252] observe that in the non-hierarchical variant, the aggregator nodes bear a higher energy burden than the non-aggregators. This pattern would also be evident in the hierarchical variant, in which lower-level aggregators are effectively sources of raw data for higher-level aggregators. This leads us to a situation in which aggregator nodes, already disproportionately important to the network application, are likely to be the first nodes to fail due to exhausting their energy supplies.

Sensor networks are often assumed to consist of large numbers of homogeneous nodes. The work of Chen et al [252] is interesting, as it suggests that a heterogenous network may be more energy-efficient. By allowing some proportion of nodes to specialise in data aggregation, this data processing burden is removed from the nodes which produce the raw source data. If we assume that the real-world phenomena to be observed by the sensornet are not subject to a uniform geographical distribution, nodes that are well-positioned to produce raw sensor data from 'interesting' areas can devote all energy and processing resources to this sensing task, perhaps being able to produce readings at a higher rate or for a longer duration than would otherwise be possible.

If this observation were to hold true in real sensor network deployments, it is perhaps not unreasonable to suggest that further specialisation is worthy of examination. Taken to the logical limit, every class of action and processing occurring within the network could be delegated to a subset of nodes dedicated specifically to this task. Whether this would yield substantive benefit to the network application is currently unclear, but it is worthy of further examination. Mhatre and Rosenberg [167] propose an approach in which there are multiple classes of nodes, each class having different capabilities and resources, utilising nodes of the more capable classes to shoulder a greater burden of the overall workload, for example by processing and aggregating the output of multiple less capable nodes. It remains to be seen if assigning specialised roles within a network of equivalent-performance nodes could provide similar benefit, offloading data processing activities to nodes that are not well placed to obtain interesting sensor readings.

Research addressing a similar problem was presented by Kotidis [125] on *snapshot queries*. In this proposal, sensornet nodes apply a local algorithm to elect a small number of nodes as representative nodes constituting a *network snapshot*. These *snapshot nodes* can provide quick approximate answers to user queries while substantially reducing the energy consumption in the network. Multi-resolution approximations could be implemented by maintaining a distinct set of representative *snapshot nodes* for different levels of data approximation. *Snapshot nodes* build compact models that capture the correlations among their measurements and those of their neighbours, with these models being simple and requiring little storage space (between a few hundred and a few thousand bytes) [125]. As the data model is small, if a *snapshot node* failure is detected it should be possible for another sensornet node to take over with the exchange of relatively little data from nearby surviving *snapshot nodes*.

It is worth noting, however, that building these correlations assumes that each raw data value will be transmitted at least once by the source node which may consume significant amounts of energy, requiring careful analysis and/or simulation to ensure that overall more energy is saved by *snapshot queries* than is consumed. The further assumption that close correlations exist between measurements taken by neighbouring nodes seems reasonable, provided that the separation distance between neighbouring nodes is sufficiently small, but this prevents the *snapshot node* approach from providing remote data replication to guard against mass failure of nodes in a given geographic cluster.

Simulations using real measurements of wind speed indicate that *snapshot queries* reduce by up to 90% the number of nodes which must participate in a user query, while providing a query result with error threshold smaller than the variance of the data values [125]. This latter point is particularly important, as any higher level of precision would represent wastage of energy, storage and network resources while providing no real benefit to the application or network operator. However, unless the *snapshot network* is maintained in a manner such that the *snapshot node* role is periodically exchanged between sensor nodes, the greater burden placed on the *snapshot nodes* will rapidly drain the energy resources of these nodes leading to a sharp reduction in network coverage.

Gao et al [78] consider the problem of aggregation in a *sparse network*. In this scenario, each node can determine from its own sensor readings whether it should participate in a given aggregation, but no entity in the network has global knowledge of where all of these *interesting* nodes are located. Typically, only a small subset of nodes have data with which to make a useful contribution, and nodes not possessing such data should conserve energy and reduce network contention by declining to participate. Gao et al show that probabilistic methods can be applied to build near-optimal ad-hoc aggregation trees in a *sparse* network, without the requirement of any centralised control and with considerable cost reduction by up to 20-30% in *push aggregation* and up to 50% in *pull aggregation*.

2.5 In-network aggregation frameworks

The first in-network aggregation framework for sensornets was the *Diffusion* framework proposed by Heidemann et al [102]. *Diffusion* operated within a *directed diffusion* data-centric routing environment, applying a data aggregation mechanism distributed throughout the data flows. Data are managed as a list of *attribute-value-operation* tuples. The *attribute keys* are simple 32-bit numbers with an assumption of out-of-band coordination of their values, and the *values* must implicitly conform to a data format relevant to the *attribute keys* against which they are assigned. The *operation* defines the interaction between data messages and *interests* expressed in the *directed diffusion* framework, generally phrased as either an *actual* matching value or a *formal* parametric comparison. Under the *Diffusion* framework, data can be aggregated to:

- *Value* - a single value summarising multiple raw data values

- *Binary value* - an event occurred or it did not
- *Area* - an event occurred within a given area
- *Application specific* - e.g. combining different types of sensed data to give a probability of an event having occurred

At each hop of the *directed diffusion* routes, a *filter* can be applied which implements in-network application-specific functionality for diffusion, signal processing, caching, aggregation, or any other task which benefits from control over data movement [102]. *Opportunistic data aggregation* requires intermediate nodes between the *source* and the *sink* to observe data flows, and can apply application-specific *filters* to apply *caching*, suppress duplicate data, or aggregate data from multiple sources at the cost of slightly increased delay. The latter option may induce a number of *nested queries*, and hence the sensornet designer must carefully balance the benefit against the energy cost.

A number of implementations of the *Diffusion* API have been implemented, including *Micro-diffusion* subset which occupies only 2050 bytes of code and 106 bytes of data, making it more suitable to extremely resource-constrained nodes than the 'full-strength' equivalent which requires 75KB of code and 24KB of data (including a support library) [102]. Although in-network data aggregation may increase latency in an application-specific manner, the reduced data volume may reduce latency if the sheer volume of data would otherwise have been time-consuming to distribute through a low-bandwidth network.

Heidemann et al state that application of aggregation in a physical testbed network of 14 sensor nodes reduced network traffic by 42% [102], in contrast to earlier simulation studies by Intanagonwiwat et al [117] which implied traffic reduction of 66-80% in a network of 250 nodes. From this we can conclude that either the traffic reduction due to aggregation increase in the size of the network, or the physical testbed captures some behaviour not captured by the simulation, or both effects are evident but confounded. If traffic reduction increases in the size of the network, this suggests that aggregation would be very valuable in a very large scale sensornet. If the simulation fails to capture physical details which affect the traffic volume to this extent, this suggests that further work is needed to validate and potentially improve sensornet simulation. However, both trials agree that aggregation reduces network traffic, albeit to a different extent, and hence could be valuable in reducing the energy cost of network activity in a sensornet.

The *Tiny Aggregation Service for Ad-Hoc Sensor Networks* (TAG) proposed by Madden et al [154] defines a framework for hierarchical in-network aggregation with a specific goal of minimising the number of messages distributed in the network to minimise latency and power consumption. TAG aggregation consists of two phases; a *distribution* phase in which queries are pushed down the network, and a *collection* phase in which aggregate values are continually routed up from children to parents. Time is partitioned into *epochs* which represent the periods in which aggregation of raw values is performed to ensure that aggregate data is available from the never-ending streams of raw data available from node sensors. Child nodes may subdivide the query *epoch* of a parent node to ensure that the child nodes (and their child nodes) deliver information from aggregation subtrees and leave sufficient time for the aforementioned parent node to perform further processing before pushing the value up to the next level of the aggregation tree.

TAG has been designed for a sensornet-like environment [154], and hence is a good match for a typical sensornet application. The integration between in-network processing and data routing within the aggregation tree allows for a range of optimisations, load-balancing, and simplifies predictions of performance. However, the aggregation tree may not be aligned with the topology of the underlying physical network or overlay network upon which it is constructed, and hence may be sub-optimal in terms of network hops expended. Also, unless the aggregation function substantially reduces the volume of data at each step, the volume of data passing each network node and link may increase substantially toward the upper levels of the hierarchy, perhaps overloading these nodes and network resources in their immediate environment.

2.6 Conclusions

Data aggregation is critical for successful sensornet systems. The unbounded volume of data produced by sensors attached to nodes must somehow be tamed. In-network aggregation enables distributed processing to share the workload between many resource-constrained nodes, and substantially reduces the volume of data traversing the network. However, aggregation necessarily discards detail to deliver its volume reductions. Future work could consider finer-detailed and more accurate tuning of the tradeoff between cost savings and accuracy, perhaps in terms of network lifetime goals and Quality of Service contracts. Better support for collaborative sensing would enable further application optimisations. Closer integration between multi-resolution aggregation and data replication could improve network resilience to mass node failure.

Chapter 3

Data-centric routing

Data routing is the set of activities through which paths are obtained through networks for the flow of data between producers and consumers. Traditional data routing mechanisms ignore the content of data, blindly forwarding packets from node to node. Although necessary for general-purpose networks, application-specific networks such as sensornets can exploit awareness of data flow content to enable various optimisations and cost reduction strategies.

3.1 Data-centric routing in sensornets

Krishnamachari defines *address-centric* and *data-centric* routing, and discuss how the application designer selects the most appropriate [14]:

- *Address-centric routing*: Each source independently sends data along the shortest path based on the route that the queries took ('end-to-end routing')
- *Data-centric routing*: Sources send data to the sink, but routing nodes en-route look at the content of the data and perform some form of aggregation or consolidation function on data originating at multiple sources.

Krishnamachari [14] considers *data-centric routing* to be synonymous with *aggregation*, although it could be argued that it is not necessary to aggregate or otherwise modify data in transit in order to make routing decisions based on the content of the data. Traditional network protocols generally ignore packet payload, utilising only metadata contained in packet headers in routing decisions [71]. Data packets generated by lower network stack layers may be wrapped as payload within higher layer packets. Under this model, network routers never consider packet payload in routing decision-making, passively forwarding data through the network unexamined and unmodified.

Active network routers examine and potentially modify passing data flows, thereby implementing part of the application [236]. All sensornet nodes are routers sharing responsibility for distributed application processing, so sensornets are *active networks. Active networks* may separate distribution of data from distribution of processing instructions to routers, or combine these with hybrid data-instruction packets called *capsules.* Router instructions may modify packet payload data, or simply instruct routers how to interpret payload data in routing decisions. For example, specific routers could be instructed to apply matching rules to packet payloads, determining next-hop by which rules hold true without changing packet payloads. Alternatively, more complex instructions could implement in-network processing of various types such as Krishnamachari's proposed aggregation [14].

Krishnamachari [14] considers three data production scenarios:

1. *All sources send completely different information (no redundancy)*: Both address-centric and data-centric algorithms will incur the same number of transmissions; no aggregation is possible.

2. *All sources send identical information (complete redundancy)*: Address-centric routing can be more efficient than data-centric routing, if the sink observes that duplicate information is arriving and issues an instruction to all but one of the sources to cease. However, this assumes much about the network and the application.

3. *Source send information with some intermediate, non-deterministic, level of redundancy*: Address-centric routing can perform no better than data-centric routing, the latter of which can reduce the overall number of transmissions by aggregating data.

It is evident that the choice of *address-centric* or *data-centric* routing is dependent on the nature of the network application; there is no single best choice for all scenarios. However, Krishnamachari et al note that *data aggregation*, and hence *data-centric routing*, is a *particularly useful paradigm for wireless routing in sensor networks* as this can eliminate redundancy and minimise transmissions, thereby conserving precious energy resources.

Govindan et al [85] observe that the static execution plan analysis methods used in typical databases may not be appropriate to sensornets queries as their costs are highly dynamic, and hence online adaptive optimisation is required. They observe that mathematically characterising the approximation quality of results and query optimisations will require statistical research rather than formal analysis, for example to consider the notion of the *eddy*, which is *a dataflow operator that is interposed between commutative query processing operators.* Based on its observations of consumption and production rates of these operators, an *eddy routing policy* can route incoming tuples to *better* operators first. Where a query spans multiple nodes it may be necessary for *eddies* to function in a distributed parallel fashion, which is currently an open research question, to continually optimise operator costs in the online query plan space.

3.2 Directed diffusion

Directed diffusion is a data-centric routing mechanism proposed by Intanagonwiwat et al [117]. All propagation, exchange and aggregation of data and queries occurs exclusively through localised interactions [118]. There is no end-to-end management of data flows, in contrast to traditional IP networks, and hence there is no need for network-scoped routing or network management mechanisms. *Directed diffusion* creates empirically good, though sub-optimal, paths through which data traverse the network from sources to sinks. Intermediate nodes along data routes can cache, transform or aggregate data, thereby offering the opportunity for conserving energy [117].

A simple *attribute-value* data naming scheme is used to describe a set of *attributes* for both *interests* and *events* in early versions of *Directed Diffusion* [117, 118]. Later versions of *Directed Diffusion* extend the *interest* model by introducing *formal operators* and data *type* descriptors to the event tuple [213].

Directed diffusion is based on a *publish-subscribe* model [118]. Users or programs *subscribe* to a set of attributes, becoming data *sinks.* Sensor nodes *publish* data that they have, becoming data *sources* [213]. Both the *publishers* and *subscribers* must share an understanding of the *value* structure if *interests* are to describe the desired information in any useful level of detail and accuracy.

The original version of *Directed Diffusion* [117, 118] implemented *two-phase pull diffusion.* In the first phase, an *interest* in an item or class of *named data* is inserted into the network at an arbitrary *sink node.* The *interest* is diffused through the network, setting up entries in *interest* tables at each node it passes. Each *interest entry* has a single gradient toward the neighbour from which the interest was received, with the specified event data rate, but does not contain any information about the *sink node* which initiated the action. The *interest entries* describe purely local data routing information, appearing at each node to have originated from an immediate neighbour.

In the second phase, when a node observes an event it sends a description to every neighbour for which it has a matching *interest entry* [117]. The field of *interest gradients* contained in the network allows information about observed real-world events to pass back through the network from *source* to *sink* along *data gradients.* It is likely in a dense network that multiple paths exist between any given pair of *source* and *sink*; positive and negative route reinforcement is used to encourage the majority of data to flow along lower-cost and lower-delay routes, but retaining the ability to fall back on less optimal routes if necessary, for example if a node involved in a low-cost route should unexpectedly fail. This is achieved by *exploratory* messages sent to neighbours; lower-latency links deliver the *exploratory* message first, and are subsequently favoured.

Silva et al [213] later observed that algorithm behaviour differs when applied within dissimilar networks. Extending *Directed Diffusion* as *one-phase push diffusion* and *one-phase pull diffusion* variants creates a family of diffusion algorithms with different performance characteristics from which network designers can select the most appropriate. *Two-phase pull diffusion* works well where a small number of sinks collect data from the network. However, it is inefficient in situations where nodes remain largely inactive until 'woken up' by neighbouring nodes to collaborate on a sensing task. *One-phase push diffusion* allows exploratory data to diffuse through the network without the establishment of *interest gradients*. It is more efficient than *two-phase pull diffusion* when the network has many sources but each source only rarely produces data, and hence the overhead of setting up *interest gradients* is too costly.

Another variant, *one-phase pull diffusion* [213], dispenses with the second *exploratory* phase of *two-phase pull diffusion*; data is delivered along the single lowest-latency path, established by the first *interest* message received at each intermediate node. This approach is potentially more energy efficient as it removes a phase in which network flooding occurs, but at the expense of producing more fragile data delivery routes.

Intermediate nodes can apply arbitrary application-specific caching, processing, rate down-conversion or aggregation to passing data flows [117].

Directed Diffusion algorithms are well-suited to the sensornet paradigm; they do not require globally unique node identifiers, utilise geographic information to form good routes, enable in-network processing and aggregation, are adaptively resilient to node failure, and do not rely on end-to-end or centralised data flow management. However, this lack of central control can be a weakness as well as a strength. As all decisions are local, with no end-to-end or system-wide control, it is possible for routes to be selected that are sub-optimal in terms of cost, delay and energy consumption balancing. *Directed Diffusion* is sensitive to the behaviour of the MAC layer [118], and hence any proposed network must be tested and analysed extensively prior to deployment to work around this fragility.

Nair and Cardell-Oliver [170] build on the work of Intanagonwiwat et al [117, 118] and Silva et al [213], presenting a formal TLA (*Temporal Logic of Actions*) specification of *Directed Diffusion*. Nair and Cardell-Oliver propose an analytical method that embraces both simulation and verification, noting that *the accuracy of wireless network simulators and their models is an open research problem*. Formal verification of an algorithm prior to simulation ensures the algorithm is correct for the intended application scope. Simulation confirms the results of formal verification, and provides designers with insight into the general behaviour of an algorithm which can then be stated as *theorems* and verified formally in the verification framework. This approach allows the designer to manage the trade-off between the *accuracy* of the *analysis model*, the *cost* of performing the analysis, and the *generality* of results obtained.

Although Nair and Cardell-Oliver [170] only present a formal specification of a small section of the overall problem, they are nevertheless able to draw some useful conclusions on the behaviour of *Directed Diffusion* by directly executing these specifications in a simulator. They find that in *push diffusion* the size of the routing tree discovered is directly proportional to the cost of diffusing data over that tree. Finding an optimal routing tree is NP-complete [14] but heuristic techniques could be developed to find a near-optimal solution. For *pull diffusion* there is no clear relationship between tree size and protocol performance, although larger trees will generally complete *pull diffusion* faster if parallel routes are permitted to develop. The lack of a clear relationship for *pull diffusion* suggests that it may be a poor choice for selection where some aspects of the deployment scenario are not known prior to deployment, or the network is expected to adaptively cope with changing conditions.

3.3 Rumor routing

Braginsky and Estrin [26] propose *Rumor Routing*, which they define as a *logical compromise between flooding queries and flooding event notifications.* In a *flooding* strategy, an item is distributed to all nodes in the network, implicitly constructing a set of possible routes from source to destination which is a subset of all possible routes. Typically, only the lowest cost routes in this set are retained, although they are not guaranteed to be optimal.

If the number of *events* observed by the network is very high compared to the number of associated *queries*, it may be useful to consider flooding queries throughout the network as the communication overhead is independent of the number of events. If the number of *events* is very low compared to the number of *queries*, it may be useful to instead flood the event notifications throughout the network as the communication overhead is independent of the number of the queries. The *Rumor Routing* algorithm is useful if the ratio of *queries* to *events* lies somewhere between these extremes [26].

Agents are packets containing a table of *events* encountered along its path, along with the hop-count to each event. *Agents* are not associated with any particular *query* or *event*, but may be created at the same time as an *event* is observed. Upon arrival at a node, the agent and node synchronise *Rumor Routing* tables and the agent time-to-live is decremented. As a passing *agent* can be received by other nodes in the radio-range of the path, the passage of an *agent* can leave a rather 'thick' trail and hence distribute a substantial amount of information.

Queries are sent on a random walk through the network until they probabilistically stumble across the *event path* of the required event, at which point the query can be routed directly to the *event* itself along the *event path*. By Euclid, two straight lines in a planar node distribution are likely to intersect. The probability of two lines interacting in a bounded rectangular region is around 69%, so if five paths to each event are constructed there is a 99.7% probability that a query path will encounter at least one of these. If this fails, the application could fall back on flooding. Geographic information can be used to encourage the packets to traverse a straight line through a dense network and hence increase the likelihood of paths crossing.

Rumor Routing relies heavily on the probability that two straight lines in a rectangular region will intersect. However, if the network instead occupies a three-dimensional region it is far less likely that straight lines through the network will serendipitously intersect. It would therefore be difficult to recommend *Rumor Routing* for use in networks organised in three dimensions such as those which might monitor a multi-floor building. Additionally, Braginsky and Estrin do not present an optimal strategy for agent generation, and suggest that the selection of an appropriate strategy or agent count is an application-dependent parameter.

3.4 Compression routing

Gupta and Kumar [90] argue that large distributed networks are not feasible, because the per-node capacity per square metre vanishes as $O(\sqrt{N})$ as the number of nodes in the network,

N, becomes large. This is a result of network congestion, a consequence of many independent data flows crossing the network. However, Scaglione and Servetto [206] argue that *the solution for sensor networks that [they] envisage exploits the same degrees of freedom that are causing the problem.* In dense sensornets, nearby nodes will yield similar data for a given observed physical phenomenon. The high level of correlation between data produced by nearby nodes is exploited by applying compression to strip out redundant information, effectively reducing the throughput per node. *Data aggregation* techniques similarly rely on redundancy within data sets, but implicitly assume the existence of redundancy without seeking specific examples. Therefore, *aggregation* techniques more reliably reduce data volume, but *compression* techniques more reliably retain data characteristics.

Non-distributed *source coding* is applied, reducing the amount of bits which must be transmitted per square metre to $O(N^{-1})$. The compression routing technique depends on the fact that, although the capacity per square metre is vanishing as the network grows ($O(\sqrt{N})$), the number of non-redundant bits generated per node is vanishing at a faster rate as the network grows ($O(N^{-1})$) [206]. Scaglione and Servetto apply classical *source codes* as sample data traverses the network, jointly re-encoding data in queues and removing redundant bits at each hop. The success of this strategy requires that the source data are distributed as Gaussian random variables, the correlation among samples is an arbitrary spatially homogeneous function, and as N grows the correlation matrix converges to a smooth two-dimensional function.

The approach outlined by Scaglione and Servetto [206] is interesting as it exploits behaviour which is typically a weakness of sensornets and extracts a positive behaviour. However, it is not without weaknesses. There is an implicit assumption that nearby nodes will necessarily yield highly correlated data, which is not necessarily true if nodes producing redundant data are simply turned off by the network energy management policy. Secondly, although the compression routing approach could be layered on any hop-by-hop routing protocol, any in-network processing or aggregation will require each intermediate node to uncompress the data before inspecting or using it. Thirdly, as with any compression mechanism, corruption of single bits in the compressed data stream may cause disproportionately large damage to the message. Fourthly, although Scaglione and Servetto provide a thorough analytical treatment of their mechanism, they do not provide empirical evidence in the form of simulations or testbed experiments. Without empirical evidence it is impossible to know if the assumptions are reasonable and that the theory is both valid and sufficiently complete.

3.5 Conclusions

General-purpose networks are typically composed of dedicated infrastructure with no awareness of the meaning of data traversing the network. Sensornets are different. Every node is a data producer, a data consumer, and a router. The entire network is dedicated to a single application. Consequently, intermediate nodes located along data routes can examine data *en route* to influence routing decisions or to enable in-network processing. Typically the latter is applied to reduce data flow volumes by compressing or aggregating multiple data points near to their source locations. Future work may continue this trend, allocating further application-specific responsibility to intermediate nodes, and provide greater support for trading-off data volume reductions against Quality of Service. Also valuable would be better management of feedback loops in sensor-actuator networks, potentially minimising latency.

Chapter 4

Peer-to-Peer networks

This chapter considers *peer-to-peer networks*, in which the standard pattern of communication is between cooperating peers acting as both clients and servers, in contrast to the traditional approach of dedicated and distinct servers and clients.

4.1 Peer-to-Peer networks

Distributed databases allow stored knowledge to be distributed across the network. However, this stored knowledge is useless if it cannot be retrieved at a later time. In *Client-Server* networks, *client* nodes consume services and information provided by *server* nodes. However, in *Peer-to-Peer* (P2P) networks, each node performs both *client* and *server* roles. *Distributed Hash Tables* (DHTs) are often employed in Peer-to-Peer networks to advertise and locate services and data.

The exact definition of 'Peer-to-Peer' is debatable [205] but the term generally is used to describe systems which *lack dedicated, centralized infrastructure, but rather depend on the voluntary participation of peers to contribute resources out of which the infrastructure is constructed.* P2P systems allow the effective utilisation of computational resources spread across a network, and are potentially robust to failures of single nodes and hence are suitable for long-term storage of data [18].

Lv et al [152] define three P2P network architecture categories:

1. *Centralised*: Centralised database of all nodes, and resources available at each node, is maintained. Poor scalability, single point of failure.

2. *Decentralised but structured*: No centralised node or resource database but topology is controlled and structured. Storage locations are managed for efficient querying. *Highly structured* systems precisely control network structure and data placement. *Loosely structured* systems use *hints* to influence, but not control, data placement.

3. *Decentralised and unstructured*: No centralised index, no precise control of network topology or file placement. Highly resilient to rapidly changing network population. Querying cannot exploit known structure, potentially generating significant traffic.

4.2 The lookup problem

P2P network storage and processing capacity is inherently scalable, but the *scalability of file location and query resolution is much more problematic* [153]. Random search methods are *inherently unscalable* [153, 152], but locating desired information often requires widespread query distribution with highly transient node populations. Inexact or widely-matching searches pose particular problems [153]. Lookups of commonly-replicated *hot* items are more successful than less-replicated *cold* items [153, 152]. Sensornet nodes with limited resources can support only limited replication, suggesting difficulty in locating information in P2P sensornets.

The *lookup problem* derives from difficulty in locating specific items within P2P networks in a scalable manner without centralised servers or hierarchy [18]. The *Gnutella* approach, flooding queries throughout all nodes, is unscalable, unreliable, and generates unacceptable traffic levels. Totally centralised database servers are unreliable and unscalable. Totally decentralised databases like *Gnutella* scale poorly as flooded requests and responses generate too much traffic. Hierarchical databases like DNS are between these extremes, but place disproportionate burden on nodes near the hierarchy root and are sensitive to individual node failures.

Scalable *overlay networks* can be constructed over base networks [18] using *Distributed Hash Tables* (DHTs). Each node bears equal data management burden, responsible for part of DHT *key space.* Producers map offered data and services to *keys.* Consumers locate desired data or services by *lookup(key)* operations against the DHT, yielding the identity of nodes providing data or services. Queries traverse networks in direction of nodes with increasingly close key. Upon receiving successful replies from producers, consumers form connections with producers by any available means, not necessarily following query paths.

DHT *Lookup algorithms* must address [18]:

- *Load-balanced key-node mapping*: Node identifiers and data keys are mapped to strings of digits using hash functions. Keys hashing to given string s are assigned to the node with *closest* string to s.
- *Forwarding key lookups*: Nodes fulfil queries if they have the requested data. Otherwise, they must forward the query to a *successor node* with ID *closer* to the hash.
- *Building routing tables*: Nodes maintain small routing tables of neighbouring nodes to determine *successor nodes.*

Scalability and low per-node requirements are desirable for sensornets. However, sensornet applications must somehow determine *keys* for desired information, requiring network-wide understanding of key composition. More fundamentally, without globally-unique node identifiers it is difficult to partition *key space* among nodes.

4.3 Data replication strategies

Data replication is not mandatory in P2P networks, but distributing copies throughout the network decreases search latency and network overhead, and increases resilience to node failure [47]. Mechanisms are required for guaranteeting equivalence of data replicas, and maintaining consistency when replicated data is updated.

P2P replication strategies include [152]:

1. *Owner replication*: *Requester* nodes replicate any information received from successful queries.

2. *Path replication*: Data replicated by all nodes along query path from *requester* to *provider*.

3. *Random replication*: Data replicated by randomly-selected proportion of nodes along query path from *requester* to *provider*.

Path replication and *random replication* outperform *owner replication* and are near-optimal in *hop count*, *average messages created per node*, *nodes probed* and *peak message density* [152].

Good replication strategies minimise *expected search size* (ESS) for *locatable* items, probing no more than a *maximum search size* (MSS) for insoluble queries [47]. Cohen proposes increased replication of commonly-requested items to reduce ESS and MSS, assuming query rate is calculable and proportional to item importance. In the *uniform strategy*, all items are replicated equally. In the *proportional strategy*, item replication is proportional to request rate. Cohen defines

> An allocation p lies between *uniform* and *proportional* if for any pair of items $i < j$ we have $\frac{q_i}{q_j} \geq \frac{p_i}{p_j} \geq 1$; that is, the ratio of allocations $\frac{p_i}{p_j}$ is between 1 (*uniform*) and the ratio of their query rates $\frac{q_i}{q_j}$, where q_n and p_n are respectively the proportion of replicas and proportion of queries associated with item n.

Proportional strategies improve common search performance at the expense of rarer searches, but surprisingly *proportional* and *uniform* ESS are identical. For allocation strategies *between uniform and proportional*, *expected search size* (ESS) is *strictly better* than *uniform* and *proportional.* For allocation strategies *outside uniform and proportional*, ESS is *strictly worse. Square-root allocation*, where the ratio of allocations for any two items is the square root of the ratio of query rates, is *between uniform* and *proportional*, and minimises ESS. Distributed replication algorithms converging on *square-root allocation* exist [47] but require more per-node state than generally feasible in sensornets.

Data replication in P2P networks overlaid on sensornets receives poor coverage in the literature, with most published work assuming Internet-like networks with large energy, storage and processing resources, without considering neighbouring nodes likelihood of hosting similar sensor data or snooping nearby wireless communications. Significantly, little consideration is given to distance between data producers and consumers, either to enable collaborative event sensing or to maximise geographic distance for redundant copies, or multiresolution replication with replica detail inversely proportional to distance from source.

4.4 Lessons from real-world networks

P2P networks are constructed as *overlay networks* using infrastructure of underlying physical networks [9]. Multiple *overlay networks* can coexist on shared physical infrastructure without application-level interaction [205]. Few studies exist on P2P network performance in sensornet-like infrastructure; most extant work assumes Internet-like infrastructure.

The *Napster* and *Gnutella* P2P *overlay networks* enable filesharing between large groups of Internet-connected hosts. *Napster* maintains centralised indexes of files shared by hosts, whereas *Gnutella* utilises fully decentralised indexes distributed using DHTs.

Although ostensibly *symmetric* rather than *client-server* networks, *significant heterogeneity and lack of cooperation across peers* exists; a significant proportion of hosts display *server-like behaviour* with the remainder displaying *client-like* behaviour [205]. In practice, some nodes enjoy higher network bandwidth, lower network latency, and greater availability. Nodes with server-like resources in highly heterogenous networks should undertake server-like roles with greater routing and content-serving responsibility [205].

In sensornets, distributing small numbers of highly-resourced nodes for server-like roles could remove significant burden from greater numbers of low-resourced nodes each incapable of supporting server-like tasks. Aligning server burden to resource availability to create *quasi-hierarchical organisations* is generally unsupported by existing P2P networks [153]. *Napster* and *Gnutella* suffer from *free-rider* nodes consuming shared resources without sharing their own in a locally-greedy manner. Aligning locally-good and globally-good decisions requires *incentives and rewards for peers to provide and exchange data* [205]. Sensornets could implement this concept by trading energy costs.

Network damage from single-node failure may increase if significant responsibility is concentrated in relatively few highly-resourced nodes. Node connectivity in large networks typically follows a scale-free power-law distribution, $P(k) = ck^{-\alpha}$ [48]. A critical fraction of nodes, p_c, can be removed without destroying network connectedness. Large networks with $\alpha \leq 3$ are very robust to random node failures. *Gnutella* has $\alpha = 2.3$ [205] and remains unfragmented until 60% of *randomly selected* nodes are removed. However, removing ¡4% of *best-connected* nodes induces serious fragmentation. Losing relatively few nodes hosting large proportions of the DHT renders a substantial proportion of DHT-indexed resources unreachable. P2P *overlay networks* in sensornets should avoid nonhomogeneity of node degree [153] to avoid fragile DHT implementations.

Gnutella query distribution employs TTL-limited flooding; computationally simple, tolerating rapidly changing network population. Selecting appropriate TTL is difficult, but adaptive *expanding ring* approaches exist [153]. TTL-limited flooding scales poorly and is resource-expensive, particularly as sensornet wireless communication is energy-expensive, occupies wireless medium for extended periods, and risks packet loss from collisions. Replacing flooding with *multiple parallel random walk* approaches significantly reduces radio transmissions [153]. k simultaneous *walker messages* reduce search delay by factor k but increases network overhead [152]. Scalable P2P searching requires *adaptive query termination*, *minimal message duplication*, and *small granularity when increasing query coverage* [152].

The *Skype* voice telecommunications introduces QoS guarantees and soft real-time requirements to P2P networking [21]. An overlay network of highly-resourced *supernodes* is maintained with every standard node associating with exactly one *supernode*, analogous to clusterhead nodes in sensornets. Traffic is routed directly between communicating users where possible. Otherwise, traffic is routed through an *overlay network* of highly-resourced *supernodes* to maintain QoS conformance [88]. Sensornets could similarly embrace heterogeneity for QoS conformance.

4.5 Constructing Peer-to-Peer networks using Distributed Hash Tables

4.5.1 Freenet

Influenced by earlier *Napster* and *Gnutella* networks, *Freenet* was the first academic P2P filesharing network based on symmetric DHT algorithms [45]. *Freenet*'s monolithic design precludes modular design principles common elsewhere in network design. *Freenet* goals include *anonymity for information producers and consumers*, *plausible deniability for access*, *resistance to information access denial*, *efficient dynamic storage and routing*, and *decentralisation of all network functions*. Most sensornet applications require only the latter two, rendering design compromises associated with anonymity preservation (and hence *Freenet*) inappropriate.

Strong anonymity also precludes observation of deployed *Freenet* [45], requiring simulation for large-scale analysis. *Freenet*'s *efficient adaptive routing algorithm pathlength scales approximately logarithmically with a change of slope near 40,000 nodes* raising scalability concerns. Anonymity requirements increase key generation complexity [45] and no *freerider* prevention exists [205].

Freenet manages discrete files only, identified by binary keys generated during file storage through hashing algorithms [45]. Nearby nodes are checked for *key collision*, the file being cached along all TTL-limited keysearch path nodes. File requesters must somehow obtain or generate appropriate keys. Queries propagate toward nodes with increasingly *near* cached keys to desired file keys, until an exact match occurs where the requested file propagates backwards along TTL-limited query path. Without an exact match, the query fails.

4.5.2 Content-Addressable Network (CAN)

CAN partitions hashtable coordinate space into *zones*, each managed by exactly one node [196]. Data keys map to coordinates, data being stored at the node whose *zone* contains these coordinates. CAN assumes zone manager nodes have sufficient storage for all zone data, and does not replicate data within zones.

Nodes leaving the network in a planned manner merge their zone with a neighbouring zone, transferring all managed $(key, value)$ pairs and changing zone shape. Sensornets may exploit this where zone-managing nodes power-down for energy management. Nodes leaving the network in an unplanned manner, such as sensor node failure, cause zone management to pass to another zone node. Zone shape remains unaltered but all managed $(key, value)$ pairs are lost. CAN efficiency requires uniform coordinate space partitioning, quickly lost if nodes frequently leave or join. CAN may perform poorly in sensornets where duty-cycling rapidly corrupts coordinate space partition uniformity, and node failure loses managed data.

4.5.3 Chord

Chord employs a one-dimensional circular coordinate space with simpler partitioning than *CAN* [227]. Each node manages *coordinate ranges* rather than *coordinate zones*, responsible for at most $\frac{(1+\varepsilon)K}{N}$ keys where N is the set of nodes and K is the set of keys. Upon nodes leaving or joining the network, $O(\frac{K}{N})$ keys are moved. Nodes maintain routing tables of $O(\log N)$ entries, and *successor lists* with unique identifiers for the next r nodes in coordinate space, allowing incremental progress in multiple node failures [18]. *Chord*'s assumptions of uniquely identified nodes may not hold in sensornets, and load-balancing support is crude. Proof-of-concept implementations of *Chord* are described in [29] and [55].

4.5.4 Pastry

Like *Chord*, *Pastry* applies a one-dimensional circular coordinate space containing uniformly distributed node identifiers [202], but offers more sophisticated coordinate space navigation, accurate storage overhead estimation, accurate performance prediction, and route length within 30-40% of optimality. Each node maintains a *routing table*, *neighbourhood set*, and *leaf set.* *Routing table* entries reference multiple candidate next-hop nodes sharing a common prefix. *Neighbourhood sets* contain *proximity* information used for routing decisions. Sensornets may usefully apply geographic distance as the *proximity metric* and utilise geography in node identification. *Leaf sets* reference nodes directly reachable by single hop for route termination. Message delivery does not require exact key matches, instead guaranteeing delivery to active nodes with nearest match; useful for sensornets with node duty-cycling and frequent failures. Implementations of *Pastry* include *PAST* storage network [64], *Scribe* multicast infrastructure [37], *SplitStream* cooperative multicast [36], and *GridKit* grid computing middleware [86].

4.5.5 Tapestry

Tapestry addresses similar goals to *CAN*, *Chord*, and *Pastry*, but is based on *Plaxton meshes* and allows applications to control data location [255] . Like *Pastry*, but unlike *CAN* and *Chord*, natural correlation between overlay topology and physical topology exists, avoiding incurrence of high physical hop counts for each logical hop. *Plaxton meshes* [184] manage node routing tables to form a large distributed set of embedded routing trees. Each network hop incrementally moves messages to nodes sharing one more digit with the required node ID, similar to *longest prefix routing* in CIDR IP address allocation [255, 184]. *Tapestry* cannot provide reliability guarantees, instead relying on efficient fault recovery [255], but algorithmic complexity renders *Tapestry* ineffective in nonstatic networks [18]. Hildrum provides useful analysis, deriving upper bound costs for common operations [108].

Tapestry is applied in *Mnemosyne* distributed steganographic storage [93] and *Bayeux* [259] fault-tolerant data dissemination systems. *Brocade* implements a second-order overlay network over *Tapestry*, itself overlaid on underlying physical networks [256]. *Brocade* exploits network heterogeneity, finding *supernodes* connected by high-capacity links as *shortcuts* to reduce latency and hop-count, similar to *KaZaA* and *Skype* [21, 88]. Sensornets typically contain many short-distance links modelled as *spatial graphs.* Introducing a small proportion ($[0.002, 0.2]$) of long-distance *shortcuts* (perhaps implemented as high-capacity wired links) converts *spatial graphs* to *relational graphs* [104], effectively converting sensornets to *small world graphs* [168]. Exploiting *shortcuts* can reduce average path length by 50%, reduce average latency, and reduce packet transmission [104].

4.5.6 Virtual Ring Routing (VRR)

VRR implements circular coordinate spaces to implement routing rather than resource location [33]. Nodes are organised in a *virtual ring* and maintain a set of nearby *virtual neighbours.* Whereas algorithms like *Chord* and *Pastry* form *query paths* to locate resources with connections completed through conventional routing algorithms, VRR instead routes data directly along *query paths.* Eliminating traditional routing algorithms reduces potential inefficiency and mismatch, effectively removing a network stack layer. This step is bold but logical, though risks some amount of network stack modularity and assumes correlation between physical network and overlay network topologies.

4.5.7 Other DHT algorithms, systems and implementations

Kademlia [164] uses *XOR metric* $d(x, y) = x \bigoplus y$ on keys (x, y) for *proximity metric* calculations, with low computational complexity, simple correctness proofs, but little correlation between physical and overlay networks.

SkipNet [95] bases distributed routing tables on *Skip Lists* rather than plain lists. Internet-like networks of backbone-connected *organisations* with predominantly interorganisation traffic enjoy performance gains, but sensornet-like flat routing hierarchies do not.

PlanetP [52] replaces *query flooding* with *query gossipping*, reducing network overhead. A distributed global document index is implemented using *Bloom Filters*, readily supporting imprecise querying with human-readable search terms. *Bloom Filters* are *optimistic*, sometimes yielding *false positives* though never *false negatives.*

PROST [185] enables *programmable structured P2P overlay networks* with dynamically loaded services exploiting reusable key-based routing infrastructure, decoupling application and infrastructure.

Ekta [189] integrates overlay network routing and physical network routing, combining *Pastry* and *DSR.* P2P networks operating in MANETs enjoy higher *packet delivery ratio*, higher *average end-to-end delay*, and lower *routing overhead*, with integrated routing than conventional layered routing. Observed benefit grows as network size grows or traffic interarrival time decreases. *Ekta* demonstrates that breaking traditional network layering and modularisation potentially yields significant performance improvements in MANETs and sensornets, though reducing flexibility, generality, and software reuse potential.

4.5.8 DHT performance

Balakrishnan [18] compares *node state* maintenance, data *lookup*, and *joining* network cost for *CAN*, *Chord*, *Pastry* and *Tapestry*, where d is network diameter and N is node count:

	CAN	Chord	Pastry	Tapestry
Node state	d	$\log N$	$\log N$	$\log N$
Lookup	$dN^{\frac{1}{d}}$	$\log N$	$\log N$	$\log N$
Join	$dN^{\frac{1}{d}} + d \log N$	$\log^2 N$	$\log^2 N$	$\log^2 N$

Operation cost is similar for all algorithms, especially if $d = O(\log N)$ as expected in uniform density sensornet deployments. Sensornet designers must instead consider other factors, such as computational complexity and performance of heuristics, for each proposed deployment.

4.6 Topology-enhanced Peer-to-Peer networks

Cramer [51] observes *DHTs and MANETs seem to be a good match*, as *both have to cope with dynamic, self-organising networks.* Single *virtual hops* in overlay networks correspond to potentially multi-hop routes in underlying physical networks. Efficient routing in overlay networks require detailed awareness of physical network topology and costs, either during overlay network construction or during routing decisions [197]. Ideally, overlay network topology would mirror physical network topology [198].

Castro [35] defines three fundamental strategies for topology-aware routing in overlay networks:

1. *Proximity routing*: Overlay network topology does not correlate to physical network topology. Next-hop nodes selection is locally-greedy, selecting either the physically closest node, or a node balancing progress in DHT keyspace against physical proximity.

2. *Topology-based node ID assignment*: Map overlay network ID space onto physical network topology, such that close overlay network IDs reference physically close nodes. Subsequent overlay routing implicitly considers physical topology.

3. *Proximity-neighbour selection*: Overlay network routing table entries contain both overlay IDs and physical distances. Routing selects next-hop with closest physical distance from set of candidates with overlay ID in desired portion of overlay ID space.

Gnutella does not consider topology correlation, generating highly inefficient routes [158] and unreachable hosts [205] in wireless networks where point-to-point connectivity is strongly correlated to geographic proximity.

T-DHT constructs DHTs where *adjacent areas in the hash table commonly have a direct link in the network*, using *network triangulation* against *reference nodes* for location discovery [131].

Ratnasamy's *binning scheme* requires nodes to independently partition themselves into disjoint *bins* where *nodes within a single bin are relatively closer to one another than to nodes not in their bin* [197]. Ratnasamy assumes topologically-close nodes display similar round-trip-time against *landmark nodes* and are placed in the same *bin*, assuming strong *latency-distance* correlation. *SAT-Match* [198] improves Ratnasamy's *binning* by adaptively adjusting node positions in the overlay network.

4.7 Geography-enhanced Peer-to-Peer networks

Ratnasamy et al [195] state that it would be *inappropriate to adopt the DHT routing algorithms [of Chord and CAN] for use on sensornets* because they typically interconnect nodes in a manner determined by logical identifiers in the DHT, which are largely independent of position in the physical network. Whereas this may be acceptable in an Internet-like network, in a sensornet this is unacceptable because neighbours in the DHT identifier space may be topologically far apart, so a single hop in the DHT-based overlay network may incur the energy cost of many packet transmissions in the underlying network.

Shenker et al [210] describe an alteration to a simple DHT-based P2P data storage network in which routing decisions are informed by geography. A modified version of the GPSR geography-aware routing protocol is used to direct packets to the geographically-closest node when storing data, which becomes the *home node* for that data. Similarly, query packets are delivered to the geographically-closest node with the requested information.

This approach was developed further by Ratnasamy et al to derive GHT, a *Geographic Hash Table*-based system for data storage in sensornets [195] using GPSR as the underlying routing system. Recognising that as the network scales in size so will the amount of data it generates, GHT is scalable, self-organising and energy-efficient by minimising communication. Stored data is replicated locally to ensure persistence in the failure of individual nodes, and load balancing is achieved by distributing load throughout the network using a geographic hierarchy.

GHT hashes keys into geographic coordinates, storing the key-value pair at the sensornet node geographically nearest to the hash of the key [195], called the *home node*. The GHT *put(key, value)* and *get(key)* operations direct packets to a geographical destination rather than a specific node. The GPSR routing mechanism was originally designed to route packets to a known non-geographic address. However, it is necessary to use a modified version of GPSR as the originator of a *put* or *get* packet does not know (and should not know) the identity of the destination node. Furthermore, it is highly likely that there is no node at the precise location specified by the hashing function. GHT defines the *home perimeter* as the ring of nodes which surround the *home node* for a given GHT packet. GPSR delivers the packet to the node nearest to the specified geographic location, the *home node*.

The *perimeter mode* of GPSR is then used to make *put* packets traverse the entire *home perimeter*, replicating the key-value pair at each node on the *home perimeter*, finally returning to the *home node* [195]. GHT's *perimeter refresh protocol* uses the replicated data stored by these *replica nodes* to allocate a new *home node* for a key-value pair if the original *home node* fails; the new *home node* may or may not be one of the *replica nodes* in the *home perimeter* of the failed node. This provides some resilience to failure of individual nodes, but does not provide resilience against *clustered failures* when all nodes in an area fail simultaneously. Ratnasamy et al [195] suggest that data could be stored multiple times at dispersed locations using multiple hash functions to address this weakness, although this seems a rather vague and inefficient proposal.

Ratnasamy et al evaluate GHT by simulation [195]. Due to the computational intractability of detailed radio modelling, detailed simulation was performed for networks of up to 200 nodes and a more abstracted simulation for networks of up to 100,000 nodes. While this simplifying abstraction allowed behaviour to be examined in a larger simulated network, removing the radio model entirely seems a poor decision for a sensornet in which high-level behaviour may be affected radically by detail at the lower levels. Future work could perhaps consider the use of a simplified radio model, and the effect this will have on the outcome.

Simulation results suggest that GHT works well in networks of stable, static nodes, offering nearly perfect availability of stored events with a 99.8-100.0% success rate in retrieving previously stored data [195]. GHT also performed well when a proportion of nodes randomly fail and restart, similar to the condition of sensornet nodes periodically powering-down to save energy. As the proportion of nodes that remained permanently available, not exhibiting this changing availability, was decreased from 100% to 0% the success rate decreased from 100% to 83.3%. This figure is impressive, and suggests that GHT could work effectively in an sensornet in which nodes conserve power in a locally greedy fashion without the need for a distributed algorithm to manage the interaction between data routing and energy management. A further set of figures shows that the success rate drops as the volatility of node availability increases for a constant ratio of up/down period for nodes that fail/restart in a cyclical fashion. If the up/down periods are 480/240 seconds the success rate is 95.7%, but if the up/down periods are 60/30 seconds the success rate drops to 75.1%. The latter is still a high success rate, but indicates that GHT performance will drop as the sensornet becomes more volatile. If delivery failure induces the sensornet application to reattempt data delivery this may expend more energy than was saved by powering-down nodes.

Seada and Helmy describe *Rendezvous Regions* (RR), a general scalable architecture for service location in wireless networks designed to cope with the limited resources and large number of nodes typical of sensornets [209]. The network topology is divided into a number of rectangular regions, each of which contains a certain density of nodes, the size of the region depending on radio range and the number of hops the network designer deems acceptable within a region. Seada and Helmy describe only rectangular regions in a flat geographical region, though it appears reasonable to extend this concept to cuboid regions in a three-dimensional space. Each region is responsible for a set of keys representing data, resources or services. Within a region, a small subset of nodes elected as *servers* provide services and store data. In a sensornet, this would allow the majority of nodes to be powered-down to conserve energy for much of the time.

The mapping of keys to regions is obtained by a hashtable-like mapping function similar to that of GHT. The key difference between GHT and RR is that GHT maps keys to *rendezvous points*, whereas RR maps keys to *rendezvous regions* [209] and can work effectively with only approximate geographical information. As routing is defined in terms of regions rather than points, RR is unaffected by node mobility *within regions* and can readily support replication within regions to ensure data remains available following failures of single nodes. However, if all nodes in a region were destroyed the data and services maintained within that region are lost. Seada and Helmy dismiss this possibility as being unlikely, although it possible to imagine that entire regions of sensornets used in disaster management applications could readily and unexpectedly disappear in response to events in the monitored environment. Another issue not addressed by Seada and Helmy is that of nodes located on the boundary between regions which fail and then restart, perhaps by powering-down and reactivating as part of an energy management policy. As only approximate geographic information is used, it may be possible for a node to power-down in region x but upon waking determine that it should be in region y, for which it would have inappropriate cached data. Another interesting possibility not considered by Seada and Helmy is for regions to overlap, as nodes in these overlapping areas would be excellent candidates for data aggregators.

Two RR *data models* are defined in terms of the *lookup-to-insertion ratio* (LIR) [209]. In a sensornet, detected events induce data insertions, whereas queries applied to stored event data induce lookups. In the *service model* the number of lookups is much larger than that of insertions, whereas the opposite is true in the *event model.* Seada and Helmy suggest that aggregation could be valuable in the *event model.* It is not clear how the data model is defined if there are broadly similar numbers of lookups and insertions. When a data insertion is required, the initiating node sends a packet to the region which manages the relevant key. The first node inside the receiving region to receive the packet, called the *flooder*, *geocasts* the packet inside the receiving region giving multiple active servers and opportunity to replicate the data. Data lookups are similar in format, but the request packet is instead delivered t just one active server in the region by *anycast*. The asymptotic total message overhead of RR is $(I \times O(\sqrt{n} + \frac{n}{R}) + L \times O(\sqrt{n}))$ and the asymptotic hotspot message overhead is $O(\frac{I}{R} + \frac{L}{min(I,R) \times S})$ where n is the number of nodes, I is the number of insertions, L is the number of lookups, R is the number of regions and S is the average number of servers per region.

Seada and Helmy conduct detailed simulations of 400-node networks and approximate simulations of 100,000-node networks and compare the performance of RR to GHT [209]. The approximate simulations discard the MAC and physical layers for reasons of computational tractability, although it might be considered risky to completely discard low-level detail in simulation of a technique designed to address an inappropriate mapping between low-level and high-level network structure. The results indicate that the normalised communication overhead of RR converges rapidly on a fixed level as LIR increases. The RR overhead is less than that of GHT for any LHT greater than around a value around 0.1, being around 80% lower for high LIR. RR is a more cost-efficient method than GHT for sensornet applications, unless the application performs fewer than 0.1 lookups per insertion in which case GHT is more efficient.

The methods described by Shenker et al [210] and Ratnasamy et al [195] are applicable only to *structured* overlay networks such as *CAN* or *Pastry*, in which data location and network topology are strictly controlled by distributed hash functions. Liu et al [147] propose an alternative location-aware mechanism suited to *unstructured* overlay networks such as *Gnutella* or *KaZaA*, in which file placement and network topology are uncontrolled and essentially random. Liu et al suggest that a large proportion of traffic in such networks is a consequence of *inefficient overlay topology* and *blind flooding*, harming scalability. Their *Location-aware Topology Matching* technique optimises overlay network topology by applying three operations:

1. *TTL2-Detector Flooding* - each peer periodically floods a TTL2-detector message to compute the cost of paths to other nodes in a hop-bounded region, applying *Slow Connection Cutting* if multiple routes are discovered to any given node

2. *Slow Connection Cutting* - suboptimal-cost connections are placed in a *will-cut list* containing peers to which the given node will not send data but will still forward traffic, eventually removing these connections completely if they have not been used after a certain delay

3. *Source Peer Probing* - if a given node has only one route to a peer, it will attempt to find another route to this peer; if an alternative route is found, only the lower-cost of the original and new routes is retained

The main benefit of *Location-aware Topology Matching* appears to be reducing traffic overhead. *Blind flooding* mechanisms induce $O(n)$ messages in an n-node network per node, giving $O(n^2)$ traffic per period if all n nodes flood. The hop-bounded flooding of *Location-aware Topology Matching* reduces this overhead to $O(n)$ but risks creating cyclic routes. Simulation by Liu et al [147] of 8,000-node overlay networks in a 22,000-node underlying network suggest that *query* *response time* and *search latency* are shortened by 62% and 55% respectively, as compared to the same overlay network without *Location-aware Topology Matching.* The average cost is proportional only to the average number of neighbours and average cost of logical links, reducing traffic cost by around 75% in simulation.

The simulation results presented by Liu et al [147] show that disabling the *will-cut list* mechanism reduces query success rate by 30-40% and the *cut-list* mechanism reduces traffic overhead by 50%. However, the latter figure refers to the reduction *after* applying the flooding-reduction mechanisms, and Liu et al do not indicate how large this 50% reduction is in comparison with the savings from reduced flooding which might be considered intuitively to be much larger. Despite these results, it does not appear that *Location-aware Topology Matching* would be sufficient to render an unstructured overlay network viable in a sensornet, because an unstructured overlay by definition still ignores the inherent structure of the underlying network. It is simply not logical to take a physical network with structure defined by geography, place on this an unstructured overlay network, and then build on this a network application which utilises the same geographical structure as the underlying physical network. It would also be necessary to simulate much larger networks to establish what proportion of improved network performance can be ascribed to flooding reduction, and what proportion to the other aspects of *Location-aware Topology Matching.*

4.8 Querying through DHTs

DHTs generally support only *exact-match lookups*, which is to be expected as they are based on a hashing function. However, far richer querying languages such as SQL based on relational algebra are widely used by application designers and allow for combinations and correlations between data. Harren et al [94] suggest that the *exact-match* functionality of hashed indexes can be used as a substrate for textual similarity searches, including both strict substring searches and fuzzy matching, by carefully constructing an index in the hash table. This approach implements a hierarchical name space on the flat identifier space of the DHT by partitioning the identifiers in multiple fields. It is then possible to support a range of standard database operators and functions including *join* and *selection* in the familiar SQL style.

To achieve the goal of *P2P query processing* with only minimal extension to existing DHT APIs, Harren et al propose a three-tier architecture [94]:

- *Local data store* - the bottom tier. This could be a filesystem, a wrapper over a database, or any other data store structure localised at each node.
- *DHT layer* - the middle layer. This must support the typical DHT *put/get* interface, but with some extensions to allow software running on a node to iterate through all data held locally on that same node.
- *Query processing* - the top layer. Contains a *query executor* which transforms application-level data-processing syntax into a sequence of DHT layer primatives.

Harren et al seek to exploit the inherent parallelism of P2P networks and to provide streamed results in the form of *online query processing*, producing a stream of perhaps imperfect answers without having to wait for the entire result set to be completed [94]. This is particularly helpful in a sensornet in which approximate answers may be the norm rather than the exception, and in which real-time (or near real-time) QoS guarantees may need to be fulfilled. Harren et al evaluate their mechanism through simulation in terms of the function $f(x)$ which describes the fraction of total result tuples received by time x for a modified version of CAN. Unfortunately, they refrain from providing experimental detail or results, but observe that the existence of *significant hotspots in all dimensions: storage, processing, and routing.* Clearly this would be unacceptable in a homogeneous sensornet in which individual nodes do not have the resources to act as hotspots. However, further work may establish whether a heterogenous network could divert hotspot activity to higher-resourced nodes, or perhaps use clusters of nodes to act collaboratively as higher-resourced nodes for this purpose. It may also be possible for future work to remove hotspots by distributing the burden more evenly, though it is difficult to reason about this without further details of Harren's work.

4.9 Conclusions

Sensornets are inherently *peer-to-peer* networks. They typically have no dedicated networking infrastructure, with each node bearing some burden of providing this infrastructure. Every node produces and consumes data in collaboration with other nodes, both near and far, working toward some collective application-specific goal. Therefore, it seems natural to select peer-to-peer networking mechanisms which are good matches for the underlying network topology, rather than working against this underlying topology and attempting to force the traditional Internet-like model. However, much existing work assumes that peer-to-peer networks are always built as overlays over the traditional Internet-like model. Future work could establish closer matching between underlying network topology and peer-to-peer overlay network topology, potentially enabling greater optimisations and energy savings.

Chapter 5

Scheduling and real-time guarantees

Real-time computing attempts to ensure that scheduled tasks begin and end at the required time in a reliable and predictable manner. Sensornets and sensornet-hosted services interact with their physical environment, both observing and influencing. End-users may depend on sensornets delivering information on the physical environment in a timely manner. The physical environment operates in real-time, therefore sensornets are necessarily real-time systems.

5.1 Real-time sensornets

Real-time systems are those in which correctness *depends not only on the logical result of the computation, but also on the time at which the results are produced* [31]. In other words, the system must produce results at the *right time*; it is not sufficient to produce results *quickly* [32]. *Real-time jobs* are single units of work that become available for execution at a *release time* and must complete by the absolute *deadline* or, equivalently, within the relative *response time* from the *release time.* A sequence of related *jobs* constitute a *task* [141]. In a *periodic task* the jobs are released with regular *period*, whereas no such constraint applies to *aperiodic tasks. Sporadic tasks* are unlike *periodic tasks* in that their minimum *release times* and maximum *execution times* are unknown *a priori* [141].

Real-time sensornet behaviour is considered by Stankovic [221], who observes that *sensor networks operate in the real world, hence timing constraints are important*, and that although some sensornets are not timing-sensitive *many sensor networks will have explicit real-time requirements related to the environment.* Some real-time requirements may apply within a network node, for example taking accelerometer readings at known intervals allows an estimate of speed to be obtained which will be incorrect if the intervals are inaccurate. Other real-time requirements will apply across multiple nodes, for example requirements to inform a base station of an observed event within a given period of the event occurring.

Stankovic observes that real-time guarantees are particularly difficult to achieve in sensornets [221] due to large scale, non-determinism, noise, and unreliability of nodes and networks. Attempts to support real-time guarantees in sensornets generally focus on scheduling the transmission of packets, and are discussed in [149, 97, 99, 139, 145] and build on previous work discussed in [128]. Liu et al [142] discuss the use of imprecise computation to enhance dependability of real-time systems, such that imprecise results are accepted if precise results cannot be calculated within the deadline. If solution quality increases monotonically with respect to time the algorithm is allowed to iterate until it terminates or the deadline is reached. In a *data-centric* sensornet, it is also necessary to consider real-time scheduling of time-sensitive network traffic [218, 4].

Real-time constraints are of particular importance where sensornest deployed to gather data for safety-critical applications, or control applications where the sensornet manipulates the environment through a feedback loop of sensors and actuators. Online dynamic desynchronisation to schedule periodic actions of nearby nodes evenly throughout time, for example to share data sampling duties or schedule sleep cycles, can be implemented using the *Desync* method described by Degesys et al [58].

5.2 Scheduling

Unless a host executes only one *task* in isolation, some form of *scheduler* is required to divide system resources among *tasks* in the time domain. A simple *cyclic executive* approach *deterministically interleaves and sequentialises the execution of periodic tasks on a CPU according to a given cyclic schedule*, but may utilise system resources sub-optimally [141].

Other *scheduling algorithms* consider job factors including *priority* and *worst-case execution time* to consider which waiting tasks should be executed at which time by a processor, of which there may be several. Typical *scheduling algorithms* include *rate monotonic scheduling*, *deadline-monotonic scheduling*, *shortest remaining time*, *least slack time*, *earliest deadline first*, and *first come first served*, although many more algorithms and variants have been proposed in the literature [141].

Scheduling analysis traditionally focuses on CPU task scheduling [212] but scheduling also applies to I/O [63], network traffic [242], middleware [207], and virtually any computing resource. *Local scheduling* manages resource schedules within isolated devices, but *distributed scheduling* algorithms are required to manage resources and tasks across distributed systems [212].

Operational systems may not consume 100% of available processing resources at all times. Some excess capacity may remain unused, perhaps by design to accommodate margins of error and small timing perturbations, to manage consumption of finite resources including energy [247], or simply because insufficient useful work exists during *slack* periods [212]. Alternatively, currently active but incompletely utilised devices may *preemptively* perform work to enable longer future *sleep periods* [83].

Operating systems may *suspend* devices between scheduled tasks and unscheduled interrupts, but rarely have sufficient visibility of higher-level application behaviour [110] to make system-optimal decisions either within devices or between distributed devices [212].

In preference to *suspending* device performance may be throttled-back to conserve energy and minimise heat production, such that processing takes longer but scheduled tasks still complete by their deadlines. These approaches implicitly assume non-linear power characteristics. Careful analysis is required to determine whether maximum efficiency is obtained running components with high performance for short periods, or low performance for longer periods. Knock-on effects on other components, schedules, task synchronisation, and QoS contracts cannot be ignored, nor can the energy consumed in implementing mode transitions. Applicable techniques include CPU frequency scaling [12, 87, 148] and voltage scaling [119, 229, 183], scheduling memory bank activation [59], and asymmetric multicore processing [127].

Complex systems may provide many services for internal and external consumption. If insufficient resources exist to schedule all possible tasks, systems must somehow prioritise and schedule the most important. *Value-based scheduling* [30] recognises that some services are more important than others, and that their relative importance varies dynamically as the system runs. Offline-generated static schedules typically use resources inefficiently, react inflexibly to failures and overloads, do not support value-added computation, and do not degrade gracefully. Sensornets are inherently poorly-resourced and unreliable. Any static schedule determined prior to deployment may quickly become inappropriate, even if optimal at the point of deployment.

Value-based scheduling [30], or *reward-based scheduling* [188], recognises the *reality that not all services have equal utility* [30]. *Utility* is the actual benefit enjoyed by a system from delivery and consumption of services, and generally cannot be known with absolute certainty. *Value* of a service is an approximation for *utility* used to influence real-time scheduling decisions. At any given time a system is assumed to be in exactly one *mode*, reflecting a particular set of related behaviours and goals. For each *mode*, offline analysis determines the set of services available for scheduling which offer the greatest *value* to the activity of the system in this *mode*. During the normal running of the system, at various decision points the system can examine its behaviour, and switch between *modes* if it determines that some other *mode* is better matched and offers better *value* than the current. Decision points may be scheduled periodically, or in response to observed events.

Offline analysis is required to determine approximations of service *value*. Although offline analysis cannot produce results as close to optimal as online analysis because of incomplete information availability, and runs the risk that analytical results become stale and increasingly poor approximations of real behaviour, online analysis is generally NP-hard and therefore incurs unacceptable overhead [30]. Methods for offline analysis of *value*, and scales against which to measure and compare *value*, based on *measurement theory* are discussed in [188].

An implicit assumption of *value-based* scheduling is that system behaviour can be partitioned into discrete *modes*. Even allowing for grouped *major modes* and *minor modes* representing similar behaviours, if there is no clear boundary between *modes* there can be no clear criteria from which to determine beneficial opportunities for *mode* transition. Some systems may have behaviour which does not readily partition into *modes*. For example, simple sensornets may contain nodes which always forward all data from neighbours and attached sensors, never changing to other patterns of behaviour, hence rendering meaningless in this context the concept of *modes*.

Although complex sensornets are likely to exhibit more complex behaviour as a result of attempting to run numerous disparate services on platforms insufficiently resources to execute all, it is unlikely that all parts of sensornets will simultaneously be performing similar tasks and transition synchronously between states. Any sensornet implementation of *value-based scheduling* must extend this concept to consider larger distributed systems in which different parts of the sensornet are in different *modes* at any given time, *mode* transitions being asynchronous between subsections, andnodes being reassigned within the network hierarchy and hence implicitly switching *mode*. A more difficult issue may be managing *mode* at the whole-system level, if system designers wish to ensure that some proportion of the network is dedicated to each necessary behaviour at any given time.

System management overhead is incurred in collecting and processing necessarily incomplete system state information, and matching this model against available *modes* to determine the best match. Further overhead may be incurred in transitioning between *modes*, an issue overlooked in the literature. System designers must consider whether induced savings outweigh these overheads. Furthermore, as it is impossible to predict the future with certainty, there exists a risk that a *mode* transition results in less-optimal, rather than more-optimal, behaviour. Emergent behaviours may become apparent as a result of unforseen or uncontrolled feedback loops, in which systems transition through *modes* in a manner unpredictable by system designers, perhaps cyclically or seemingly randomly.

Feedback loops are intentionally introduced by *feedback control scheduling*, based on *feedback control theory*, and intended for *open environments where both load and available resources are difficult to predict* [224]. Lu [150] observes *while open-loop scheduling algorithms perform well in predictable environments in which the workloads can be accurately modelled . . . they can perform poorly in unpredictable environments*, yielding an *extremely expensive and underutilized system.*

Under *feedback control scheduling* [150], *manipulated variables* represent system attributes dynamically changeable by the scheduler, *controlled variables* represent measurable system attributes, and *performance references* defined desired system behaviour in terms of *controlled variables*. The *feedback control loop* consists of a *monitor*, *controller*, and *actuator*. The *monitor* measures *controlled variables*, feeding samples to the *controller*. The *controller* compares *controlled variables* against *performance references*, calculating required changes in resource utilisation. The *actuator* implements changes in permitted resource utilisation for tasks in concert with the underlying *basic scheduler*.

Sensornets are data-centric systems in which the processing activity to be scheduled may depend on the content of raw data, workloads have uncertain arrival patterns, and complex interactions distributed across multiple nodes abound. Coarse distributed models of system state are constructed dynamically. Online scheduling is influenced by the content of these models, and consequent schedules influence system behaviour by pushing this in desirable directions. Ideally, system behaviour would converge toward optimal steady-state behaviour. However, designers must tradeoff speed of convergence against behavioural stability, in effect *damping* the feedback loop to prevent undue influence of transient conditions on long-term behaviour. If the environment changes very quickly, the *feedback control scheduling* loop may not converge quickly enough to keep pace. If environmental conditions are cyclical, necessary feedback damping may induce *hysteresis* effects, system behaviour changes lagging behind environmental changes.

Substantial offline analysis is required to understand the effects of tuning system parameters for effective online decisions, assuming that it is possible for effects induced by changing parameters to be well understood. The *FECSIM* simulator provides proof-of-concept support tools for such analysis [224]. Although Stankovic et al [224] and Lu et al [150] see convergence toward steady-state, or at least convergence toward slowly-moving target behaviour, as the desired outcome, other outcomes are possible in feedback systems. The system may oscillate cyclically between behaviours, perhaps as a result of *hysteresis*. Poorly-tuned feedback loops may quickly push system behaviour into unpredictable and unhelpful states, if the magnitude of induced change grows unbounded with each iteration of the *feedback loop*. Also noteworthy is that feedback mechanisms, effectively searching the potential schedule space, may force system behaviour into suboptimal *local minima*.

5.3 Quality of Service

If a *hard real-time task* misses its deadline this may cause *catastrophic consequences on the environment under control*, whereas if a *soft real-time task* misses its deadline although this is not desireable for performance reasons there is no serious damage to the environment and this *does not jeopardize correct system behaviour* [32]. *Hard real-time* behaviour is generally defined in terms of *hard deadlines*, whereas the *soft deadlines* of *soft real-time systems* may be defined as *Quality of Service* characteristics.

Sensornets necessarily interact with the physical world and are therefore *real-time systems*. Whether a given sensornet is *hard real-time* or *soft-real time* is system-specific, though typical underlying wireless networks cannot support *hard real-time* guarantees [97].

If monotonic solution quality improvement is not observed, it is possible to schedule separate tasks for a precise algorithm with unknown computation time and an imprecise algorithm with known computation time [143]. In one variant, the precise algorithm task is allowed to execute until the remaining time before the deadline is equal to the computation time of the imprecise algorithm task (assuming that the precise algorithm task does not terminate before this), at which point the imprecise algorithm task is scheduled to guarantee that at least an imprecise result is available at the deadline.

Alternatively, the imprecise algorithm task could execute first and run to completion, the precise algorithm task is then scheduled and allowed to run until the deadline is reached, at which point the solution quality is examined to determine which result to use [144]. This latter approach risks wasting processor time on the imprecise result if not eventually used, but allows the precise algorithm task to be abandoned at any time with guaranteed availability of the imprecise result, to allow processor resources to be reassigned elsewhere. Algorithms to schedule tasks in this manner exist with cost $O(n^2)$ and $O(n^3)$ in the number of tasks. However, this overhead is unacceptable in sensornets with extreme resource constraints, especially if there are many tasks, as efficiency gains will not outweigh increased scheduling overhead.

5.4 Conclusions

Sensornets are real-time systems by virtue of their necessary interactions with the real physical environment which proceeds in real-time. However, it is generally difficult to guarantee real-time behaviour in sensornets, as underlying technologies and network layers do not provide guarantees on which to base guarantees of higher-level behaviour. Furthermore, without full control over the physical environment, it is impossible to fully guarantee conformance to real-time requirements.

Analytical methods to derive *absolute guarantees* are unlikely to be appropriate to sensornets. The large scale and complexity of sensornets renders many analytical methods impossible in this context. More fundamentally, the inherently uncertain and unknowable behaviour occurring within sensornets and interactions with the physical environment precludes the derivation of any useful absolute guarantees. For example, it is impossible to guarantee that any given pairwise internode radio link will successfully pass any information. Assuming all radio links behave similarly, it is not possible to guarantee any successful data flow whatsoever within the sensornet. No entirely accurate measurement of the physical world is possible; measurement discovers ranges of possible values within which the actual values lie.

Even if unrealistic simplified models are considered, the resulting *absolute guarantees* may be so pessimistic as to be of no practical use [22]. However, sensornet designers may usefully consider *probabilistic* or *statistical guarantees*, perhaps derived by extending the work of Bernat et al [22] for distributed timing analysis using *copulas*. *Probabilistic* or *statistical guarantees* are likely to be sufficient for the purpose of sensornet applications operating from data with inherent uncertainty. If we accept that *absolute guarantees* are not achievable in any real system, *statistical guarantees* offer genuinely useful insight into system behaviour. Sensornet designers can analyse and manage uncertainty to determine probability of correct behaviour in a given environment, perhaps using these probabilities as goals of the design process.

Future work could consider including geographical data, to ensure that sensornet activity happens both at the right time and the right place, and Quality of Service tradeoffs between real-time behaviour and energy-aware behaviour.

Chapter 6

Cross-layer challenges

In any networked computer system there are necessarily interactions between services and components. It is possible to identify recurring themes or concerns, and regions of overlap. Issues discussed in this chapter cut across multiple network stack layers.

6.1 Localisation

Localisation is required in sensornets to assign location context to sensor data, and to enable application-independent services such as *geographic routing* and *geographic storage* [237]; it is *a prerequisite to the utility of most sensor networks* [11]. If fitting a GPS unit to each node is infeasible, perhaps for reasons of cost, size, deployment area (e.g. underground) or energy consumption [91], a compromise is to have nodes infer their position from a small proportion of *seeds* which are aware of their location at all times [203]. Location inference mechanisms can be divided into the two categories of *range-based* and *range-free* [203]. *Range-based* techniques attempt to measure the distances between nodes, and include methods based on *received signal strength*, *time of arrival*, *time difference of arrival*, *angle of arrival*, and *Monte Carlo approaches.* *Range-free* approaches include *triangle intersection regions*, *hop-count distance estimation*, *Bèzier curve estimation*, and *radiointerferometric methods.* Sensornet link generation should utilise Euclidian distance bias, typically ignored in much network research [233].

6.2 Mobility

The *mobility* model of a dynamic network in which node location varies with time warrants special attention in a sensornet. A trade-off exists between energy consumption and accuracy. *Localisation* activity consumes energy and should be minimised to maximise network lifetime, but performing *localisation* too infrequently runs the risk of location data becoming stale and inaccurate and hence harming all network activities which rely on location [237]. The *Monte Carlo Localization* (MCL) algorithm [112] addresses this problem, and in fact relies on node mobility to function at all. MCL only works effectively with low node speeds, but the related *MSL* and *MSL** algorithms degrade more gracefully with node speed at the expense of greater computational complexity [203].

6.3 Energy management

Energy awareness and *energy management* are themes running throughout most aspects of sensornet design and operation. The energy resources of nodes are typically small and cannot be replenished after deployment, so energy consumption is *the most important factor that determines sensor node lifetime* [190]. Raghunathan et al observe that energy optimisation for sensornets is complex as it *involves not only reducing the energy consumption of a single sensor node but also maximising the lifetime of an entire network*, requiring dynamic trade-offs between *energy consumption*, *system performance*, and *operational fidelity* [190], yielding *up to a few orders of magnitude* of improved lifetime. Of course, if nodes can *scavenge* sufficient energy from their environment, network lifetime could extend indefinitely.

The radio module is usually the most power-hungry component of a sensor node. Surprisingly, operating the radio in *idle* mode does not provide any advantage in power consumption, and receiving data may consume more energy than sending data [190]. Therefore, energy savings must come throughout the protocol stack, influencing the design and operation of sensornet applications, networking protocols, network topologies and network tasking. Sending a single bit of information 100m may consume more energy than 1000-3000 CPU instructions [54, 186], a cost incurred at both the sender and receiver node, and by any intermediate nodes in multi-hop paths which may grow as the network becomes larger. Therefore, energy savings must be driven by energy-aware design throughout the network stack, rather than relying on improvements in hardware technology [190].

Received signal power in sensor networks drops off rapidly with distance, as r^4 due to partial cancellation from ground ray reflection with short antenna heights [186, 68]. An interesting consequence of this non-linear relationship is that network routes containing many short-distance hops may be more energy-efficient than routes containing few long-distance hops, albeit at the expense of requiring more intermediate nodes to be awake to forward traffic. Hop count, a commonly used cost metric in conventional routing protocols like RIP [100], may be inappropriate in sensornets requiring better-suited replacements considering physical factors to be found.

6.4 Security

Security is important in sensornets, as they may be deployed to provide safety-critical information in hostile environments. For example, in military applications it is feasible that opposing forces could implement some form of attack. Sensornet security requirements are categorised by Stajano and Anderson as *availability*, *authenticity*, *integrity*, and *confidentiality* [220]. Zhou and Haas [257] add *non-repudiation* to these requirements. Routing mechanisms are particularly vulnerable [257, 160, 176, 258], and communications *anonymity* may be required to prevent an attacker gaining insight by correlating patterns of real-world events and sensornet communications [258].

Sensornets are inherently insecure, as communications in the wireless medium are subject to *snooping* and *interference*, and sensornet algorithms generally assume that nodes will cooperate and follow the network rules correctly [114]; naturally, an attacker is under no such restrictions. Wireless link error rate is broadly comparable to that of wired links, despite external noise, unless specifically attacked [66]. An attacker may attempt to hijack or reverse-engineer a node, particularly if they have physical access to the network [114], or install their own spoofed nodes to subvert the network [178, 62, 172, 114]. The limited resources of sensornet nodes render traditional security mechanisms difficult to implement [180], and indeed creates a new class of attacks in which sensornet nodes are maliciously tricked into wasting energy [220]. Data collected and processed in real-time by sensornet applications may need to be secured in real-time [124]. Sensornet security frameworks include *SPINS* [180], *TinySec* [122], and *MiniSec* [151].

6.5 Resource-constrained platforms

Sensornet nodes may require specialised *operating systems* which consume minimal resources while providing the services required by sensornet applications [20]. The *TinyOS* operating system, libraries and APIs [109], and the *nesC* language in which they are implemented [80], are considered a defacto standard [163, 165, 40]. However, other sensornet-focused operating systems exist, including *SOS* [92], *RETOS* [40], *MANTIS* [2], MagnetOS [20] and Contiki [65]. It is conceivable that extremely resource constrained platforms would have no conventional operating system at all, and larger high-capacity nodes could run conventional operating systems such as *VxWorks*, *Embedded Linux*, *Windows CE*, *LynxOS*, *VRTX* or *QNX*. As sensornet deployments diversify, case-by-case design decisions will determine which operating system (if any) is appropriate to the application and hardware platform as during sensornet design. Selecting lightweight operating systems allows allocation of more resources to application-specific processing tasks. Sensornet designers must consider the tradeoff between selecting the most lightweight operating system which can support the application without overhead of unneeded services, and practical considerations of cost, development time, and testing time.

In heterogeneous sensornets it is feasible that multiple operating systems would be required to cooperate. *Middleware* provides a mechanism through which these dissimilar platforms can interact [201]. Wolenetz et al consider the requirements of sensornet middleware [251], concluding that it must support component-based development of dynamically reconfigured data fusion applications on low-resource wireless platforms. Henricksen and Robinson [107] categorise sensornet middleware systems as being *database inspired*, *tuple space solutions*, *event-based solutions*, or *service discovery based*. Middleware systems designed specifically for sensornets include *MiLAN* [103], *Impala* [146], *Tenet* [82], *TinyCubus* [159], *EnviroTrack* [1], *Cougar*

[23] and *TAG* [154]. Other relevant middleware implementations include *RUNES* [49], *3DMA* [73], and *Prism-MV* [157]. However, little consideration has been given to interoperation with traditional middleware stacks managing processing at highly-resourced base stations.

6.6 Standardisation

Standardisation would enable greater reuse of sensornet design elements, decrease development time, and allow greater interoperability with other networks. Consensus does not currently exist in the community as to the precise purpose, direction, content, or certification of any proposed standard. Indeed, there is no agreement on which body would specify or enforce any such standards. However, some exploratory work has been published by Culler et al [53] and Toh et al [240]. Standardisation of potentially any system element is possible to some extent, other than deployment-specific tasking and optimisation. Mirroring traditional platforms, greater standardisation is possible toward the lower network stack layers. Standardised APIs, protocols, and service description languages, enable interoperability within sensornet frameworks where specific deployment factors require selection of specific component instances from available alternatives.

6.7 Interoperability

Interoperability of sensornet networking protocols is of concern when connecting two or more sensornets, or when connecting a sensornet to a dissimilar network such as the Internet. Some argue that sensornets are unlike the Internet and hence require specialised protocols to replace *Internet Protocol* (IP) [186] such as *Sensornet Protocol* (SP) [53], requiring gateways for network interconnection [161]. Others argue that IP can be successfully implemented in low-power wireless networks [219] if modified appropriately. For example, the *6LoWPAN* IETF working group seeks to run *IPv6* over *IEEE802.15.4* networks [129]. Hybrid approaches such as *Tenet* have been proposed [82] in which an IP backbone connects regions in which a specialised protocol is used, analogous to the internet backbone connecting *autonomous systems.*

6.8 Design support

Software development for sensornet applications is *currently a cumbersome and error-prone task since it requires programming individual sensor nodes, using low-level programming languages, interfacing to the hardware and the network, only supported by primitive operating system abstractions*, revealing a *strong need for programming abstractions that simplify tasking sensor networks* [200]. Römer proposes that *middleware* offers a solution [200]. Other significant work in improving the maturity and quality of sensornet software development includes the definition of *design patterns* [79], and support for *interface contracts* that can be checked both statically and dynamically [10].

Sensornet design and evaluation frequently requires *simulation* and *emulation*; large-scale *testbeds* or *field trials* are infeasible and costly [17, 228, 231]. Formal analysis of sensornets may [113] or may not [6, 199, 193] be feasible. Experimenters must determine acceptable accuracy-scalability tradeoffs [56, 50, 192]; simulation-derived results are meaningless if simulated behaviour does not sufficiently match real behaviour [38, 181] and are particularly sensitive to timing discrepancies [130]. Wireless communication models are usually the component with highest computational cost [171] but the greatest source of inaccuracy [27, 126, 101].

Discrete event simulations are well suited to computer network simulation [15]. *Simulation models* [17] are constructed, similar to those used in model checking [217], and executed in *simulation engines* [84]. Incorporating real application code, execution environments, hardware, network connections, or other real entities, into *simulation models* yields *emulation models* [84], improving accuracy but harming scalability [241]. Real and simulated entities interact directly in *hybrid simulations* [166, 249, 134].

Numerous sensornet-relevant simulators and emulators exist. Unfortunately, no current examples offer total accuracy or reach desired scalability. *TOSSIM* offers cycle-accurate low-level emulation of Berkeley motes running TinyOS but very simplistic network modelling [138]. *ns-2*, the predominant network simulator in sensornet research [138, 27], uses highly-detailed network models [27] but is single-threaded [105] and scales only to around 100 simulated nodes [171]. *GloMoSim* exploits parallel execution of multithreaded simulation, scaling to 10000 simulated nodes across 10 processors [16]. Interentity communication across parallel simulation hosts [199] negates benefit of additional processors by Amdahl's Law [8, 239]. Simulating sensornet-scale networks of millions of nodes requires entity-concatenation [15] and layer-concatenation [194, 27] to reduce memory footprint [38], further sacrificing simulation accuracy.

6.9 Scalability

The number of nodes deployed in a sensornet may be of the order of hundreds or thousands, or in extreme cases of the order of millions [7], hence *scalability* is a critical issue. Emergent properties may be manifest in large deployments that do not become evident in smaller deployments. It is infeasible to individually configure every node as there may simply be too many, and more fundamentally their precise location is unlikely to be known before deployment [74]. Therefore, it may be useful to address sensornets at a level of abstraction higher than the individual nodes. Welsh and Mainland describe *abstract regions*, defined in terms of radio connectivity or geographic location, as a high-level building block for sensornet programming; code written against *abstract regions* will be automatically *compiled down to the detailed behaviour of individual nodes* [248]. Frank and Römer describe techniques to define *roles* which can be automatically assigned to nodes in a self-configuring sensornet, on the assumption that the behaviour of nodes will not be uniform throughout the network at all times [74].

The *Kairos macroprogramming system* [89] allows the programmer to express the desired *global behaviour* of the entire sensornet, automatically generating code to express the *local behaviour* of nodes. The *Regiment macroprogramming system* [173] similarly allows the entire sensornet to be programmed and tasked as a single entity, automatically *deglobalizing* high-level descriptions of required behaviour into node-level, event-driven programs. Systems such as *TinyDB* [155] allow a distributed database to be queried and programmed as a single entity at run-time, rather than compile-time as in *Kairos* or *Regiment*. Reprogramming deployed sensornets could use *application specific virtual machines* [136] such as *Maté* [137] to reduce the size of code to be distributed and interpreted in virtual machines on sensor nodes.

6.10 Distributed data models

Data models define the data representation and semantics shared by all data model users [85]. Raw sensor observations occupy a 3-dimensional spatial-temporal datacube of coordinates *x-coordinate*, *y-coordinate*, and *time* [75].

Shenker [210] describes network cost of canonical distributed data models for sensornets of n nodes:

1. *External storage*: Event descriptions delivered to external storage at cost $O(\sqrt{n})$. In-network queries and responses cost $O(\sqrt{n})$. Externally-generated queries and responses have zero network cost; all information is already external to the network.
2. *Local storage*: Event descriptions stored locally at producing node at zero network cost. Query flooding costs $O(n)$ for internally- and externally-generated queries. Responses delivered to query source with cost $O(\sqrt{n})$.
3. *Data-centric storage*: Event descriptions, labelled by *name*, stored at arbitrary network location at cost $O(\sqrt{n})$. Query delivery and response both cost $O(\sqrt{n})$ for internally- and externally-generated queries.

Data-centric storage is preferred for sensornets with many nodes, or detecting many events where many remain unqueried. *Local storage* is preferred where most detected events are queried [210].

Although data elements may be distributed throughout sensornet nodes, logical entities called *virtual tables* allow users to address distributed data as a single, location-transparent entity [85]. Queries against sensornets typically involve *aggregation* or *internode correlation*, often applied to streamed data within temporal and geographic boundaries [23]. *BigTable* [44] implements the *virtual table* model and supports large-scale parallel computation through *MapReduce* [57] but demands resources generally unavailable within sensornets.

Distributed file systems such as GFS [81] are generally unsuited to streaming data typical of sensornets. Distributed databases specifically targetting sensornet-like applications, architected as *virtual databases* [85], including *COUGAR* [23] and *TinyDB* [155] are required as *sensor queries* do not readily map to traditional relational databases [23] lacking dedicated servers [233].

Sensornet data, not sensornet nodes, require unique identification [210]. *Named data* are categorised by *type* and *location* [210] attributes *external* to network topology, *relevant* to applications, *independent* of network topology, and utilised throughout network stack layers [102].

Sensor readings are *measurements not facts*, corresponding to continuous probability distributions rather than exact values [23]. *Approximate* queries and joins against *formal parameters* rather than *exact matches* are required [102]. *Intentional Naming System* [5] and *Linda* [34] support *approximate queries*, whereas X.500 [182] and LDAP [254] do not.

6.11 Reducing data volume

Shenker et al [210] opine that *the overwhelming volume of observations produced by [network node] sensors is both a blessing and a curse*, producing unbounded volumes of data [60] with insufficient network and energy resources precluding forwarding all data to base stations for processing [210]. Localised algorithms perform *in-network processing* to minimise data flow volume [60].

Data compression algorithms reduce data volume by exploiting structure or redundancy [230]. General lossless compression algorithms utilise *run-length encoding* or *Huffman coding* [115, 24], *arithmetic coding* [250, 24], *adaptive coding* [46], and *universal coding* [261]. Lossy compression methods for streaming time-signal data typical of sensornet applications [75] include finding *base-target signal correlation* [60, 61], and *wavelets* [3, 230, 214, 42]. Wavelets readily enable *approximate query processing* [42] as implemented in *Dimensions* [75].

A sensornet is *a proxy for a continuous real-world phenomenon* which necessarily *samples that phenomenon discretely at some rate, with some degree of error* [85]. *Multi-resolution storage* stores information at different abstraction levels, using more storage where higher precision is required. *Data aging* progressively decreases data precision and hence storage overhead over time, enabling *graceful degradation* assuming that newer data are more valuable to network operators than older data. High-quality query responses are produce for recent data, and lower-quality responses are still available for older data [76]. Application-dependent data aging algorithm fine-tuning is recommended [77].

Other approaches reduce sensor node raw data production, enabling energy savings from powering-down nodes not required for data collection. *Network snapshots* exploit redundancy within sensor observations of nearby nodes [60], electing a small proportion of *representative nodes* queried through *snapshot queries* [125]. *Acquisitional Query Processing* [155] enables finer-grained control, collecting sensor data only in direct response to user queries, but consequently precluding retrospective data mining [60].

6.12 Forecasting

Time sequence data forecasting attempts to predict future data values from existing measurements, finding application in [41]:

- Interpolating for missing values or periods in stored data streams
- Discovering periods when nodes can power-down with low probability of missing interesting events
- Calculating what proportion of data streams must be retained to retain key characteristics

Forecasting linear data sequences can use *Moving Average*, *Autoregressive*, or *Moving Average-Autoregressive hybrid* models [41]. Many real-world sequences are nonlinear, however. No single method works well with all nonlinear sequences, but where sequences are *periodic* or *chaotic* rather than *random* application designers may select an appropriate method. *Artificial Neural Networks* can *learn* characteristic behaviour, *Hidden Markov Models* can be extracted from real-world dynamic systems, *Method of Analogues* applies only to *low-dimensional chaotic attractors* over short periods, and *F4 Delay Coordinate Embedding* applies only to sequences with *fractal dimension.*

Related methods examine *self-similar* [234, 158] and *multifractal* [243] patterns in network *burstiness*, *flow density* and *bytes transmitted per unit time* [43]. Applications include detecting abnormal activity or failures, congestion control, and Quality of Service management by scheduling traffic appropriately [135, 233, 43]. Overlaying multiple *bursty* data streams counterintuitively increases *burstiness* rather than smoothing-out data flow, with repercussions for managing peak data flow periods [135]. Synchronisation of chaotic systems is non-trivial, but careful design may yield *self-synchronising* systems without need of external forced coordination. Systems whose behaviour naturally converges on desired behaviour require less management, with consequent savings in energy, network bandwidth, and processing overhead. *Direct Sequence-Code Division Multiple Access* (DS-CDMA) [225] applies these principles within the *Data Link Layer*, but future work could reapply these principles within other layers or combinations of layers.

6.13 Modelling sensornets

Models can be described in terms of *inputs*, *outputs*, *parameters*, *relationships*, *mappings to and from the real world*, *model limitations*, and *model reliability* [211]. *Prediction systems* [72] consist of *mathematical models* with *prediction procedures* for determining unknown parameters and interpreting results. Unfortunately, required model inputs are often not *measurable or even directly observable, in any useful engineering sense* [211].

Paulk paraphrases Box: *All models are wrong; some models are useful* [179]. Models are an *abstraction of reality* [211] or an *abstract representation of an object* [72]. Models cannot provide perfect representations of real entities, but can be close enough to yield meaningful and useful results. Calibration significantly improves model accuracy, but models derived solely from post-hoc analysis of particular data sets perform poorly [72]. Models should undergo both theoretical and empirical evaluation before considering results trustworthy [211].

Models may be designed specifically for each system, or fixed models reused for classes of similar systems [72]. Translation between different models of a given system is frequently problematic and subjective, but useful results are possible if models display *tolerable correspondence* [96].

Govindan et al [85] argue that sensornets require *novel data-centric routing mechanisms* and *reconsideration of traditional network and database interface layering.* Govindan [85] states *it might be necessary to collapse layers or selectively break abstraction boundaries for efficiency or robustness reasons* because *communication using [sensor node] radios requires significantly more energy than computation.* Their reasons for breaking abstractions and strict layering are laudable and understandable, and not without precedent; consider multi-layer network switching [204].

However, we should be cautious when discarding notions of abstraction and indirection found critically import elsewhere in Computer Science; *separation of a system into relatively independent modules with clearly defined interfaces is a hallmark of good design* [208]. Another interpretation, contradictory to Govindan [85], is that existing research has not yet discovered a *suitable* model or set of abstractions for reused across components, applications and networks, and which is sufficiently robust and efficient.

Traditional layered network models include the OSI model [260] and the TCP/IP model [162]. Govindan [85] modifies the OSI model, proposing a sensornet model with layers for *hardware devices*, *radio and MAC*, *packet delivery*, *local signal processing*, *data-centric routing*, *collaborative event processing*, and *applications.* Tilak [238] defines a higher-level, unlayered sensornet model of *network architecture*, *performance metrics*, *communication patterns and mechanisms*, *data delivery*, and *network dynamics.* Akyildiz [7] extends the OSI model with orthogonal planes of *power management*, *mobility*, and *task management*, representing issues cutting across all network layers. An obvious ommission of these models is modelling of the environment in which the sensornet resides, including the physical phenomena to be measured by sensor nodes and other factors which are not to be measured but will nevertheless influence the operation of the network [133, 7].

Shenker models sensornet data as [210]:

- *Observations*: Single, raw, low-level sensor readings
- *Events*: Constellations of related *observations* of defined *event type*
- *Web of events*: Hierarchy of events, some defined in terms of other events
- *Event notifications*: Messages describing *events.* omitting raw *observations*
- *Tasks*: Instructions to nodes describing required data gathering

- *Actions*: Activities undertaken by nodes upon detecting *events*
- *Queries*: User or application requests to elicit *event* information

Nested queries allow nodes receiving *primary queries* to issue *secondary queries*, allowing greater abstraction in end-user queries [102, 60].

6.14 Conclusions

Modularisation, componentisation and layering attempt to reduce dependencies and interactions between system elements. However, some issues cut across the entire sensornet system, and consequently these cross-layer issues require cross-layer solutions. Although cross-layer issues may derive from functional requirements, a greater proportion derive from non-functional requirements or the design process.

Chapter 7

Conclusions and future work

This chapter reviews the findings of previous chapters. Building from these discussions, potential directions and themes for future research are discussed. Initial experimental work is proposed as a starting point for the remainder of the project, with brief discussion of later work to build on the results of these initial experiments.

7.1 Material covered

Sensornets are data-centric networks in which *knowledge management* is central to both successful network management and end-user applications. Traditional store-and-forward models in which all physical data collected by dumb sensors is transported unprocessed to base stations do not scale to realistic sensornet deployments, and would overwhelm network, processing, storage, and energy resources of small sensornet nodes. New models and methods are needed for data, metadata, queries, storage, and processing.

Some level of in-network processing is required to render sensornet applications viable, principally by reducing data volume and exploiting parallelism. Data streams are processed along network routes, each intermediate node contributing processing resources. Distributed algorithms operating along routes must make local decisions with globally-positive consequences, with minimal knowledge of distant network state. Sensornets are typically application-specific, thus enabling content-specific algorithms to exploit awareness of specific data sets, and enabling context-specific algorithms to exploit awareness of nearby network conditions.

Quality of Service requirements imply necessary tradeoffs between timeliness, accuracy, energy consumption, reliability, and other factors. Complex interactions between these factors are poorly understood in general, particularly as influenced by changing environmental conditions. Better support is required for adaptive behaviour to maintain conformance to QoS guarantees in changing circumstances.

Currently, the literature provides an uneven coverage of sensornet topics. Some subjects have attracted a body of work of such scale that, although not precluding further advances by future work, suggests that the research community may wish to focus its efforts on other matters. For example, numerous multi-hop routing protocols have been proposed in the literature, but middleware operating at a higher level receives surprisingly little coverage.

Solutions to some sensornet problems may be obtained by taking work published in other computer networking fields and adapting this work to the specific characteristics, capabilities and environments of sensornets. It is equally important to note that much of the work published in the sensornet domain may be of wider interest if modified to be more general. It is hoped that the proposed work discussed below will yield results relevant to the sensornet domain, but can be modified to be more generally applicable. For example, sensornet system management algorithms may be genericised and reapplied in other computing domains pertaining to distributed cooperative systems. Successfully reapplying theory in different domains helps to build confidence in the validity and correctness of the theory.

7.2 Requirements of initial research

Initial research work should:

1. **Produce results and insight**: First and foremost, the initial work should provide results. While it is unlikely that early results will enable groundbreaking new insights, it should nevertheless produce results which can be analysed to gain insight into the subject under consideration rather than duplicate previous experiments.

2. **Assess viability of topic**: If initial work suggests that further study of this topic is infeasible, requires resources that are not available, requires skills beyond those of the researchers, requires time beyond that available, or seems unlikely to yield new or interesting findings, then it may be necessary to pursue an alternative topic. Ideally, this new topic would be closely related to the existing topic, and would allow reuse of results delivered in the initial work.

3. **Identify further work**: Initial work should provide insight as to which research directions are likely to yield interesting results, and which will be feasible within the scope of the project.

7.3 The problem

The theoretical *value* of a network grows with the number of nodes, n. Quadratic, exponential, and $n \log n$ *value growth* models have been proposed, but most researchers agree that *value growth* is superlinear in n [28]. Sensornets may scale to millions of nodes [7], consequently of much higher theoretical *value* than simply the sum of their parts compared against noncooperating dumb collection systems.

However, harnessing this *value* remains difficult. Operators require sensornets to fulfil high-level functional and Quality of Service requirements, but network scale and changing environmental conditions preclude operator micromanagement of nodes or node resources. Network-level requirements must somehow filter down to individual nodes. Nodes have incomplete visibility of network status, but must make local decisions with global consequences. No single node has the resources to address the entire network-level problem, but can address part of the problem and contribute part of the solution. System-wide coordination is required to ensure the composite system performs all required functions where individual system components cannot.

Sensornets must necessarily pursue *aggressive energy management* policies [98], encompassing all entities and functions of the network and application [222]. Existing energy management research typically applies oversimplified models and cannot provide adequate network lifetime guarantees [98]. *Aggressive energy management* is not achieved by passively accepting tasks sets and schedules, and determining the most energy-efficient way to execute these. Instead, energy-awareness must permeate all aspects of system design and management, actively influencing task sets and schedules to minimise energy consumption, perhaps sacrificing some operational accuracy within user-tolerable bounds. Minimal energy consumption becomes a first-order goal for consideration in system management tradeoffs. These policies may manage energy at different system granularities, for example per-system, per-node, or per-subsystem. System elements should be powered-down whenever possible, should scale back energy consumption when they must be active, and should minimise energy-hungry component state changes.

More generally, *aggressive resource management* is required for effective use of limited storage, processing power, network bandwidth, and all other finite resources. Some resources, for example battery-delivered energy, can be used only once and not replenished. Some resources, for example RAM, can be reallocated to numerous tasks provided allocations do not overlap temporally. Resources cannot generally be transferred between nodes, but responsibility for resource production or consumption can be transferred. Sensornets management may control node activity and resources in terms of availability *density* within regions with lower overhead than managing nodes individually. Node responsibility definitions may be in terms of *roles* assigned to nodes, leaving nodes to act autonomously within role boundaries. Managing role assignment spatially and temporally remains an unsolved problem.

Complex interrelationships between resources may exist. For example, increasing node RAM usage may require activation of additional memory banks, thus increasing energy consumption. Interrelationships between resource availability and the physical environment may exist. For example, solar-powered nodes cannot scavenge energy in the dark, hence power management policies may consider daylight hours and moon phase. Raw sensor data may be available only during limited periods. Real-world interactions necessarily require sensornet design to consider geographic and realtime issues.

Resource availability and resource demand is distributed spatially and temporally in sensornets. Effective sensornet function requires adequate resources to be available at the right place and time to fulfil system-level requirements. Executing a given task at a given location at a given time requires certain resources, which perhaps cannot be determined in advance, to be available simultaneously.

Within the scope of this project it is not feasible to complete all work described below, much less solve completely the issue of *energy-aware adaptive sensornet system management*. However, these proposals raise useful and interesting research questions. Initial work will establish which of these possible directions represents the most promising avenue for future in-depth investigation. As work progresses, research scope will progressively reduce as the direction and feasibility of future work becomes more clear. The goal of this initial work is to derive and define a specific, appropriate, and novel research problem of appropriate scale to be addressed during the remainder of the PhD.

7.4 Addressing the problem

7.4.1 System management policies

Much published work considers resource allocation in either the spatial or temporal domain, but surprisingly little work attempts to unify these. Spatio-temporal optimisation may be necessary to enable maximal system efficiency and efficacy. Search and optimisation techniques could explore phase space in which resource supply and demand, hence production and consumption, are embedded against time.

It may be necessary to trade-off energy consumption against sensornet fidelity if the predicted cost of sensornet activity is too high to sustain the desired level of accuracy, perhaps requiring *graceful degradation* of service if conformance to *Quality of Service* contracts cannot be reconciled with energy budgets. However, it may be possible to improve sensornet energy management by exploiting application-dependent optimisations. Whereas application-dependent optimisations may be undesirable in a general purpose network, as sensornets are inherently application-specific these opportunities should not be ignored.

A general theme of recent sensornet research is the move upward within the network stack when seeking efficiency improvements, in some case suggesting solutions that cut across multiple network stack layers. Interlayer activities must be regulated by appropriate control mechanisms and policies to ensure layers cooperate harmoniously, rather than in mutual opposition. For example, it may prove beneficial to coordinate or synchronise sleep periods of processors and networking components. Otherwise, if at some given time one component is active and the other inactive, the active component cannot perform useful work and wastes energy while maintaining its active state. Although effective in other system classes, static cross-layer coordination cannot work effectively in sensornets adaptive to their changing environment.

Maximising sensornet efficiency may require new system management policies and algorithms which consider all available knowledge. Users may require sensornet management to target different goals, for example *maximum lifetime*, *maximum detail*, *time-bounded maximum detail* or some other compromise. Finding optimal system configurations before deployment reduces management overhead during post-deployment network activity. Online adaptive methods converging on near-optimal solutions may enable sensornets to remain useful in changing and adverse conditions. Assuming networks are deployed with good starting configuration, and must adapt to changing conditions, effective rules are required to determine when and how to change behaviour. Changes may be continuous, for example changing sampling rate, or discrete, for example changing behaviour mode from raw data collection to processed data forwarding. Optimal rules defining behaviour change are application-specific, and must be established or evolved using offline analysis.

It is interesting to consider the effect of applying localised algorithms to large networks in which multi-hop routing is required, as the local algorithm may or may not be successful in coordinating local activity when the influence of more distant nodes is factored in. Alternatively, a distributed approach could be employed, perhaps within sensornet node clusters or in a hierarchichal network structure. Local decisions have global consequences, perhaps requiring some nodes to make locally-poor decisions for the greater good of the network.

7.4.2 Distributed models

Effective sensornet management may require construction and maintenance of resource supply and demand maps, distributed throughout the sensornet. Consideration must be given to historical state, current state, and forecasted future state. Fully accurate resource maps are infeasible due to high overheads, measurement uncertainty, inconsistency as information updates navigate the network, and impossibility of predicting the future. Approximated mapping, perhaps reducing resolution with distance from information source or hierarchically, reduces overheads but offers reduced certainty the further an application looks, either spatially or temporally.

Maintaining a multiresolution approximate network state model, distributed within the network itself, requires each node to store and process model data corresponding to itself, its neighbours, and distant nodes. Neighbour information could be obtained by:

- **Passive snooping** of nearby packet transmissions. For example, it might be known that the sensornet application switches off a node as soon as it has transmitted a packet. In this case, a node X observing a packet transmission by node Y could deduce that Y is about to switch off and is therefore unsuitable as a recipient for another packet at this time. Alternatively, if it is known that Y will listen after transmitting, this suggests an ideal opportunity to schedule packet transmission $X \rightarrow Y$.
- **Active transmission of activity data**, in which each node appends a summary of its model its own activity and that of its neighbours to some or all outgoing packets. This avoids unnecessary radio state transitions, but runs the risk that more energy is wasted on transmitting unneeded data than is saved by the application of needed data.

- **Active request of model data**, in which a node requests activity data from some or all of its neighbours. Some protocol for query submission and result packaging would be required, with consideration of energy overhead in design.

Postprocessing and inference rules allow extraction of further knowledge from this data. Notably, absence of certain information may allow inference of other information.

7.4.3 Scheduling

Scheduling may offer one approach to reducing the energy cost of sensornet activity. Each node is responsible for local application tasks such as sensing its physical environment and processing locally-produced data, distributed application tasks such as data aggregation and collaborative sensing, and network tasks such as routing and forwarding data. These tasks may be *periodic* or *sporadic*, and each implies that one or more nodes must be active at the requisite time.

It is generally impossible to predict the future. Sensornets are likely to be non-deterministic networks interacting with a non-deterministic environment. Nevertheless, it is reasonable to assume some degree of statistical structure will be evident, for example resulting from periodic sensor polling or radio beaconing. This structure could be exploited to inform in-network decision making to coordinate activity, using either local or distributed algorithms.

Whereas traditional scheduling analysis considers *periodic* or *sporadic* tasks, real-world phenomena may exhibit more complex behaviour. Temporal distribution of real-world events may be *chaotic*, fundamentally different to *random* or *periodic* distributions. Scheduling policies attuned to characteristics of physical phenomena producing raw data that drive the system may enable greater efficiency than possible in sensornet systems ignorant of such patterns. Currently it is not known whether sensornet activity should be driven solely by occurrence of real-world phenomena, solely by in-network scheduled periodic activity, or some combination thereof. Online adaptivity and optimisation requires closed feedback loops of sensornet self-monitoring, learning, and self-adjustment.

Nodes sustain energy costs when transitioning between states or maintaining active states. Intelligent scheduling can reduce energy consumption. Energy is wasted by radio transmissions if the intended recipient is not awake or is not listening; coordination between sender and receiver activity cycles could reduce the quantity of energy wasted in this manner. It may be energy-efficient to execute a processing task with low CPU clock speed for a longer period than to execute the same task with high CPU clock speed for a shorter period, if this would not overrun the task deadline. If multiple packets are to be transmitted by a node via radio, it may be possible to batch multiple packets for a single transmission if this would not overrun deadlines for any of these packets, hence reducing energy-hungry state transitions.

There exists a large body of work on task scheduling, and a significant body of work on network scheduling, but there appears to be little published work which seeks to unite the two. This union could permit application-specific optimisations in which software running on nodes could build an approximate model of their neighbours activity and their own activity. A further union with geographical management provides interesting opportunities for optimisation through multivariate search techniques. In-network processing as data traverses the network implies accompanying resource demands also traverse the network simultaneously, requiring distributed scheduling of resource production and consumption, sensing tasks, processing tasks, and node roles, in both temporal and spatial domains.

Pairwise internode communication requires that two nodes within wireless range be active simultaneously. Sensornet applications generally require communications between two unique locations rather than two unique nodes, typically implying multi-hop routes. Distributed control mechanisms may be required to ensure sufficient density of active nodes along communication routes for the volume of data traversing the network. Control mechanisms reduce the cost of managed activities, but incur additional management costs which must be outweighed by savings for net gain.

Coordination between the activity cycle of nodes could permit convergence toward a near-optimal pattern within regions. A local algorithm could be obtained by applying the principles of the *DESYNC* algorithm [58]. This could be self-adapting to network change, for example in the case of node failure. The coordination algorithm should converge to a network state in which each node is awake for minimal time, and is awake only when required to perform useful activity. For example, if node X is required to perform a calculation, transmit the results to Y, then Y performs further processing, a naive solution would require both X and Y to wake together, perform these activities, then shut down. A more energy-efficient solution could be for X alone to wake, perform its calculation with Y waking just as this completes, the transmission from X to Y completing, then X switching off while Y performs its calculation.

7.5 Initial work

Simple statistical models could be built of nodes, the network through which they interconnect, and the environment in which they reside. Lightweight simulation models enable scalable simulations, and deriving models from real experiments or high-cost high-accuracy simulations enables confidence in result accuracy. Starting with very simple models, system complexity is increased progressively by increasing sophistication of simulated event injection and building additional system layers above those already present. At each stage, observation at per-node, regional, and whole-system scales examines behavioural patterns, trends, and interactions. Online adaptivity can be investigated if time allows by adding simple in-network monitoring, learning, and response systems, augmenting existing scheduling and network management mechanisms.

Analysis could consider the content of the per-node activity model summarising the activity of the local node and its neighbours. Investigation would be required to establish which data should be retained in trading-off size against accuracy. Some data may be more valuable than others, being better predictors of future behaviour. Factors to consider include timescales over which to retain historical data, diminishing value of data with passing time, diminishing value of data with geographic and network distance, network management statistics, task management statistics, and physical phenomena statistics. Multi-resolution representations could be considered, for example using more storage space for more accurate representations of recent events and nearby nodes. Different data representation models may be beneficial for in-network system models, and external models.

7.6 Further work

Further work may consider the interplay between scheduling algorithms for tasks, node state, and network activity, operating within and between sensornet nodes, and geography. Local and distributed algorithms could be considered, to determine if the higher cost of a distributed algorithm is outweighed by any higher savings in energy consumption. Modifications to existing sensornet scheduling algorithms may be required to utilise knowledge of neighbouring-node activity, to enable distributed energy-aware system management. Any impact of these modifications to node behaviour on Quality-of-Service characteristics would be measured. Research could consider trade-offs between Quality of Service and energy cost, for example to calculate the most energy-efficient configuration that conforms to a given Quality of Service contract, or to provide a graceful degradation of Quality of Service as the energy budget is reduced.

Later work could consider the possibility of discovering a set of good initial values to assign to parameterised behaviours prior to deployment by applying search techniques and using simulation to calculate the fitness function. If the simulation model closely matches the real environment and network, the resultant proposed configuration may behave in a near-optimal fashion when deployed. This may minimise the time for the network to converge on a steady near-optimal state, and minimise the amount of network activity to achieve coordination. Another possible experiment would be to complete the feedback loop by extracting status information from the network and pushing improved tasking back into the network. Online simulation could be used to explore changes to network tasking, provided that it executes faster than real-time, and a fitness function used to determine if network behaviour is improved by any of a set of possible changes.

Sensornet networking is inherently peer-to-peer, but much existing work seeks to build Internet-like overlay networks poorly matched to physical network topology. Improved integration and mapping between DHT keyspace, geographic regions, and hierarchical network structure, could enable more efficient peer-to-peer sensornets with network function divided between regions in a manner sympathetic to actual physical structure. Dividing the physical deployment area into regions corresponding directly to regions of multidimensional DHT keyspace would be one approach. *Directed diffusion* could use DHT methods to publish sources and sinks, and to establish routes.

It is impossible for any simulation model to achieve complete accuracy, particularly for a changing environment. However, a sufficiently complete and accurate model could be explored prior to sensornet deployment using offline simulation running on highly-resourced platforms with complete knowledge of the simulation model, rather than relying solely on online adaptation and the greatly limited knowledge at each node.

One approach to determining which details are significant and should be retained in the simulation model, and which details are insignificant and could be discarded, is by running a series of simulations in which detail is progressively stripped from the model. An experimental search-based methodology for deriving optimal algorithm configurations based on *censoring*, in which algorithm responses are measured only to a specified threshold, described by Poulding et al [187] could be modified for a sensornet context. Evaluating the quality of simulation results for a given approximate model, perhaps by comparing deviation of output from that of the most complete model, enables the experimenter to determine which factors have the greatest influence on result accuracy. The factors found to be most significant are those to be retained when reducing the complexity, and hence execution expense, of the simulation model.

7.7 Conclusions

Wireless Sensor Networks, or *sensornets*, are composed of large numbers of minimally-resourced nodes, cooperating and communicating wirelessly. *Wireless Sensor Networks* are typically deployed to monitor the physical environment into which they are deployed, though *Wireless Sensor-Actuator Networks* complete the feedback loop by influencing or modifying the environment they observe.

Wireless Sensor Networks show great promise and flexibility, have been investigated by the academic and industrial research community for a number of years, and have been proposed for a wide variety of environment monitoring problems. Despite these strengths, however, the number of real deployments in production environments is surprisingly low. One possible interpretation is that, despite the diversity of high-quality work published to date, some pieces of the jigsaw remain missing.

Although *Wireless Sensor Networks* share some properties of conventional wired computer networks, they differ fundamentally in some respects; they are *energy-aware*, *geography-aware*, *data-centric*, *application-specific*, *self-configuring* and must be *highly scalable*. Although it is desirable to reuse as much theory and accumulated knowledge from existing network research as possible, there remains a need for *sensornet-specific* solutions to *sensornet-specific* problems.

Bibliography

[1] T. Abdelzaher, B. Blum, Q. Cao, Y. Chen, D. Evans, J. George, S. George, L. Gu, T. He, S. Krishnamurthy, L. Luo, S. Son, J. Stankovic, R. Stoleru, and A. Wood. Envirotrack: Towards an environmental computing paradigm for distributed sensor networks. In *ICDCS '04: Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, pages 582–589, Washington, DC, USA, 2004. IEEE Computer Society.

[2] H. Abrach, S. Bhatti, J. Carlson, H. Dai, J. Rose, A. Sheth, B. Shucker, J. Deng, and R. Han. Mantis: system support for multimodal networks of in-situ sensors. In *WSNA '03: Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 50–59, New York, NY, USA, 2003. ACM Press.

[3] F. Abramovich, T. Bailey, and T. Sapatinas. Wavelet analysis and its statistical applications. *J. R. Statist. Soc. D*, (48):1–30, 2000.

[4] Maria Adamou, Sanjeev Khanna, Insup Lee, Insik Shin, and Shiyu Zhou. Fair real-time traffic scheduling over a Wireless LAN. In *RTSS '01: Proceedings of the 22nd IEEE Real-Time Systems Symposium (RTSS'01)*, page 279, Washington, DC, USA, 2001. IEEE Computer Society.

[5] William Adjie-Winoto, Elliot Schwartz, Hari Balakrishnan, and Jeremy Lilley. The design and implementation of an intentional naming system. In *SOSP '99: Proceedings of the seventeenth ACM symposium on Operating systems principles*, pages 186–201, New York, NY, USA, 1999. ACM Press.

[6] J. Ahn and P. Danzig. Speedup vs. simulation granularity. *IEEE/ACM Transactions on Networking*, 4(5):743–757, October 1996.

[7] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks (Amsterdam, Netherlands: 1999)*, 38(4):393–422, 2002.

[8] Gene M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. *Readings in computer architecture*, pages 79–81, 2000.

[9] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, and Robert Morris. Resilient overlay networks. In *Symposium on Operating Systems Principles*, pages 131–145, 2001.

[10] Will Archer, Philip Levis, and John Regehr. Interface contracts for TinyOS. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 158–165, New York, NY, USA, 2007. ACM Press.

[11] Joshua N. Ash and Lee C. Potter. Robust system multiangulation using subspace methods. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 61–68, New York, NY, USA, 2007. ACM Press.

[12] Hakan Aydin, Rami Melhem, Daniel Mossé, and Pedro Mejìa-Alvarez. Power-aware scheduling for periodic real-time tasks. *IEEE Trans. Comput.*, 53(5):584–600, 2004.

[13] Ryan Aylward and Joseph A. Paradiso. A compact, high-speed, wearable sensor network for biomotion capture and interactive media. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 380–389, New York, NY, USA, 2007. ACM Press.

[14] S. Wicker B. Krishnamachari, D. Estrin. Modeling data-centric routing in wireless sensor networks. In *Proceedings of the INFOCOM 2002*, 2002.

[15] R. Bagrodia and R. Meyer. PARSEC: A parallel simulation environment for complex system. *IEEE Computer*, pages 77–85, October 1998.

[16] Lokesh Bajaj, Mineo Takai, Rajat Ahuja, Rajive Bagrodia, and Mario Gerla. Glomosim: A scalable network simulation environment. Technical Report 990027, University of California at Los Angeles, 13, 1999.

[17] Sandeep Bajaj, Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, Padma Haldar, Mark Handley, Ahmed Helmy, John Heidemann, Polly Huang, Satish Kumar, Steven McCanne, Reza Rejaie, Puneet Sharma, Kannan Varadhan, Ya Xu, Haobo Yu, and Daniel Zappala. Improving simulation for network research. Technical Report 99-702, USC Computer Science Department, March 1999.

[18] Hari Balakrishnan, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Looking up data in P2P systems. *Commun. ACM*, 46(2):43–48, 2003.

[19] Daniel Barbara. Mobile computing and databases - a survey. *Knowledge and Data Engineering*, 11(1):108–117, 1999.

[20] Rimon Barr, John C. Bicket, Daniel S. Dantas, Bowei Du, T. W. Danny Kim, Bing Zhou, and Emin Gün Sirer. On the need for system-level support for ad hoc and sensor networks. *SIGOPS Oper. Syst. Rev.*, 36(2):1–5, 2002.

[21] Salman A. Baset and Henning Schulzrinne. An analysis of the Skype peer-to-peer Internet telephony protocol. Technical Report CUCS-039-04, Columbia University, September 2004.

[22] Guillem Bernat, Alan Burns, and Martin Newby. Probabilistic timing analysis: An approach using copulas. *J. Embedded Comput.*, 1(2):179–194, 2005.

[23] Philippe Bonnet, Johannes Gehrke, and Praveen Seshadri. Towards sensor database systems. In *MDM '01: Proceedings of the Second International Conference on Mobile Data Management*, pages 3–14, London, UK, 2001. Springer-Verlag.

[24] Abraham Bookstein, Shmuel T. Klein, and Timo Raita. Is Huffman coding dead? In *SIGIR*, pages 80–87, 1993.

[25] Jonathan Bowen. The ethics of safety-critical systems. *Communications of the ACM*, 43(4):91–97, 2000.

[26] David Braginsky and Deborah Estrin. Rumor routing algorthim for sensor networks. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 22–31, New York, NY, USA, 2002. ACM Press.

[27] Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, John Heidemann, Ahmed Helmy, Polly Huang, Steven McCanne, Kannan Varadhan, Ya Xu, and Haobo Yu. Advances in network simulation. *Computer*, 33(5):59–67, 2000.

[28] Bob Briscoe, Andrew Odlyzko, and Benjamin Tilly. Metcalfe's Law is wrong. *IEEE Spectrum*, pages 22–31, July 2006.

[29] Emma Brunskill, Frank Dabek, Frans Kaashoek, David Karger, Robert Morris, Ion Stoica, and Hari Balakrishnan. Building Peer-to-Peer systems with Chord, a distributed lookup service. In *HOTOS '01: Proceedings of the Eighth Workshop on Hot Topics in Operating Systems*, page 81, Washington, DC, USA, 2001. IEEE Computer Society.

[30] A. Burns, D. Prasad, A. Bondavalli, F. Di Giandomenico, K. Ramamritham, J. Stankovic, and L. Strigini. The meaning and role of value in scheduling flexible real-time systems. *J. Syst. Archit.*, 46(4):305–325, 2000.

[31] Alan Burns and Andy Wellings. *Real-Time Systems and Programming Languages*. International Computer Science Series. Pearson, Harlow, 3rd edition, 2001.

[32] Giorgio Buttazzo. *Hard Real-Time Computing Systems*. Springer, New York, 2nd edition, 2004.

[33] Matthew Caesar, Miguel Castro, Edmund B. Nightingale, Greg O'Shea, and Antony Rowstron. Virtual ring routing: network routing inspired by dhts. *SIGCOMM Comput. Commun. Rev.*, 36(4):351–362, 2006.

[34] Nicholas Carriero and David Gelernter. The S/Net's Linda kernel. *ACM Trans. Comput. Syst.*, 4(2):110–129, 1986.

[35] M. Castro, P. Druschel, Y. Hu, and A. Rowstron. Topology-aware routing in structured peer-to-peer overlay networks. Technical Report MSR-TR-2002-82, Microsoft Research, Microsoft Research, One Microsoft Way, Redmond, WA 98052, 2002.

[36] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: High-bandwidth content distribution in a cooperative environment. In *Proceedings of (IPTPS'03)*, February 2003.

[37] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in communications (JSAC)*, 20(8):1489–1499, 2002.

[38] David Cavin, Yoav Sasson, and André; Schiper. On the accuracy of MANET simulators. In *POMC '02: Proceedings of the second ACM international workshop on Principles of mobile computing*, pages 38–43, New York, NY, USA, 2002. ACM Press.

[39] Alberto Cerpa, Jeremy Elson, Deborah Estrin, Lewis Girod, Michael Hamilton, and Jerry Zhao. Habitat monitoring: application driver for wireless communications technology. *SIGCOMM Comput. Commun. Rev.*, 31(2 supplement):20–41, 2001.

[40] Hojung Cha, Sukwon Choi, Inuk Jung, Hyoseung Kim, Hyojeong Shin, Jaehyun Yoo, and Chanmin Yoon. RETOS: resilient, expandable, and threaded operating system for wireless sensor networks. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 148–157, New York, NY, USA, 2007. ACM Press.

[41] Deepayan Chakrabarti and Christos Faloutsos. F4: large-scale automated forecasting using fractals. In *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, pages 2–9, New York, NY, USA, 2002. ACM Press.

[42] Kaushik Chakrabarti, Minos N. Garofalakis, Rajeev Rastogi, and Kyuseok Shim. Approximate query processing using wavelets. In *VLDB '00: Proceedings of the 26th International Conference on Very Large Data Bases*, pages 111–122, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[43] D. Chakraborty, A. Ashir, T. Suganuma, G. Mansfield Keeni, T. K. Roy, and N. Shiratori. Self-similar and fractal nature of internet traffic. *Int. J. Netw. Manag.*, 14(2):119–129, 2004.

[44] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: A distributed storage system for structured data. In *OSDI'06: Seventh Symposium on Operating System Design and Implementation, Seattle, WA, November, 2006*, pages 205–218, 2006.

[45] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. *Lecture Notes in Computer Science*, 2009:46, 2001.

[46] J. G. Cleary and I. H. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, COM-32(4):396–402, April 1984.

[47] Edith Cohen and Scott Shenker. Replication strategies in unstructured peer-to-peer networks. *SIGCOMM Comput. Commun. Rev.*, 32(4):177–190, 2002.

[48] Reuven Cohen, Keren Erez, Daniel ben Avraham, and Shlomo Havlin. Resilience of the internet to random breakdowns. *PHYS.REV.LETT*, 85:4626, 2000.

[49] P. Costa, G. Coulson, and C. Mascolo. The RUNES middleware: A reconfigurable component-based approach to networked embedded systems. In *Proceedings of (PIMRC05), IEEE, Berlin, Germany, September 2005*, 2005.

[50] J. Cowie, D. Nicol, and A. Ogielski. Modeling 100,000 nodes and beyond: Self-validating design. In *DARPA/NIST Workshop in Validation of Large Scale Network Simulation Models*, 1999.

[51] Curt Cramer and Thomas Fuhrmann. Proximity neighbor selection for a DHT in wireless multi-hop networks. In *P2P '05: Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing (P2P'05)*, pages 3–10, Washington, DC, USA, 2005. IEEE Computer Society.

[52] Francisco Matias Cuenca-Acuna, Christopher Peery, Richard P. Martin, and Thu D. Nguyen. Planetp: Using gossiping to build content addressable peer-to-peer information sharing communities. In *HPDC '03: Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC'03)*, page 236, Washington, DC, USA, 2003. IEEE Computer Society.

[53] D. Culler, P. Dutta, C. T. Ee, R. Fonseca, J. Hui, P. Levis, J. Polastre, S. Shenker, I. Stoica, G. Tolle, and J. Zhao. Towards a sensor network architecture: Lowering the waistline. In *Proceedings of HotOS X: Tenth Workshop on Hot Topics in Operating Systems*, June 2005.

[54] David Culler, Deborah Estrin, and Mani Srivastava. Overview of sensor networks. *IEEE Computer*, 37(8):41–49, 2004.

[55] Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Wide-area cooperative storage with CFS. In *Symposium on Operating Systems Principles*, pages 202–215, 2001.

[56] Judith S. Dahmann, Richard Fujimoto, and Richard M. Weatherly. The Department of Defense High Level Architecture. In *Winter Simulation Conference*, pages 142–149, 1997.

[57] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. In *OSDI*, pages 137–150, 2004.

[58] Julius Degesys, Ian Rose, Ankit Patel, and Radhika Nagpal. DESYNC: self-organizing desynchronization and TDMA on wireless sensor networks. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 11–20, New York, NY, USA, 2007. ACM Press.

[59] Victor Delaluz, Mahmut Kandemir, N. Vijaykrishnan, Anand Sivasubramaniam, and Mary Jane Irwin. Hardware and software techniques for controlling dram power modes. *IEEE Trans. Comput.*, 50(11):1154–1173, 2001.

[60] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Data reduction techniques for sensor networks. Technical report, University of Maryland, July 2003.

[61] Antonios Deligiannakis, Yannis Kotidis, and Nick Roussopoulos. Compressing historical information in sensor networks. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 527–538, New York, NY, USA, 2004. ACM Press.

[62] John R. Douceur. The sybil attack. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 251–260, London, UK, 2002. Springer-Verlag.

[63] Fred Douglis, P. Krishnan, and Brian Marsh. Thwarting the power-hungry disk. In *USENIX Winter*, pages 292–306, 1994.

[64] Peter Druschel and Antony Rowstron. PAST: A large-scale, persistent peer-to-peer storage utility. In *HOTOS '01: Proceedings of the Eighth Workshop on Hot Topics in Operating Systems*, page 75, Washington, DC, USA, 2001. IEEE Computer Society.

[65] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *LCN '04: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN'04)*, pages 455–462, Washington, DC, USA, 2004. IEEE Computer Society.

[66] David Eckhardt and Peter Steenkiste. Measurement and analysis of the error characteristics of an in-building wireless network. In *SIGCOMM '96: Conference proceedings on Applications, technologies, architectures, and protocols for computer communications*, pages 243–254, New York, NY, USA, 1996. ACM Press.

[67] Ali Ozer Ercan, Abbas El Gamal, and Leonidas J. Guibas. Object tracking in the presence of occlusions via a camera network. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 509–518, New York, NY, USA, 2007. ACM Press.

[68] D. Estrin, L. Girod, G. Pottie, and M. Srivastava. Instrumenting the world with wireless sensor networks. In *Proceedings of ICASSP 2001, Salt Lake City, Utah*, May 2001.

[69] Deborah Estrin, Ramesh Govindan, John S. Heidemann, and Satish Kumar. Next century challenges: Scalable coordination in sensor networks. In *Mobile Computing and Networking*, pages 263–270, 1999.

[70] C. Farrar, H. Sohn, M. Fugate, and J. Czarnecki. Integrated structural health monitoring. In *Proceedings of SPIE's 8 th Annual International Symposium on Smart Structures and Materials, Newport beach, CA*, March 2001.

[71] S. Feit. *TCP/IP*. McGraw-Hill Series on Computer Communications. McGraw-Hill, New York, 2nd edition, 1996.

[72] N. Fenton. *Software metrics: a rigorous approach*. Chapman Hall, London, 1991.

[73] Tore Fjellheim, Stephen Milliner, Marlon Dumas, and Kim Elms. The 3DMA middleware for mobile applications. In *Proceedings of Embedded and Ubiquitous Computing, International Conference EUC 2004*, pages 312–323, 2004.

[74] Christian Frank and Kay Römer. Algorithms for generic role assignment in wireless sensor networks. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 230–242, New York, NY, USA, 2005. ACM Press.

[75] Deepak Ganesan, Deborah Estrin, and John S. Heidemann. Dimensions: why do we need a new data handling architecture for sensor networks? *Computer Communication Review*, 33(1):143–148, 2003.

[76] Deepak Ganesan, Ben Greenstein, Denis Perelyubskiy, Deborah Estrin, and John Heidemann. An evaluation of multi-resolution storage for sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 89–102, New York, NY, USA, 2003. ACM Press.

[77] Deepak Ganesan, Ben Greenstein, Denis Perelyubskiy, Deborah Estrin, and John Heidemann. An evaluation of multi-resolution storage for sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 89–102, New York, NY, USA, 2003. ACM Press.

[78] Jie Gao, Leonidas Guibas, Nikola Milosavljevic, and John Hershberger. Sparse data aggregation in sensor networks. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 430–439, New York, NY, USA, 2007. ACM Press.

[79] David Gay, Phil Levis, and David Culler. Software design patterns for TinyOS. In *LCTES '05: Proceedings of the 2005 ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems*, pages 40–49, New York, NY, USA, 2005. ACM Press.

[80] David Gay, Philip Levis, Robert von Behren, Matt Welsh, Eric Brewer, and David Culler. The nesC language: A holistic approach to networked embedded systems. In *PLDI '03:*

Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation, pages 1–11, New York, NY, USA, 2003. ACM Press.

[81] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google File System. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 29–43, New York, NY, USA, 2003. ACM Press.

[82] Omprakash Gnawali, Ki-Young Jang, Jeongyeup Paek, Marcos Vieira, Ramesh Govindan, Ben Greenstein, August Joki, Deborah Estrin, and Eddie Kohler. The tenet architecture for tiered sensor networks. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 153–166, New York, NY, USA, 2006. ACM Press.

[83] Richard A. Golding, Peter Bosch II, Carl Staelin, Tim Sullivan, and John Wilkes. Idleness is not sloth. In *USENIX Winter*, pages 201–212, 1995.

[84] E. Goturk. Emulating ad hoc networks: Differences from simulations and emulation specific problems. In *New Trends in Computer Networks. Volume 1 of Advances in Computer Science and Engineering Series*, 2005.

[85] R. Govindan, J. Hellerstein, W. Hong, S. Madden, M. Franklin, and S. Shenker. The sensor network as a database. Technical report, University of Southern California, Computer Science Department, 2002.

[86] P. Grace, G. Coulson, G. Blair, L. Mathy, W. Yeung, W. Cai, D. Duce, and C. Cooper. GRIDKIT: Pluggable overlay networks for grid computing. In *Proc. International Symposium of Distributed Objects and Applications (DOA'04), Larnaca, Cyprus*, October 2004.

[87] Dirk Grunwald, III Charles B. Morrey, Philip Levis, Michael Neufeld, and Keith I. Farkas. Policies for dynamic clock scheduling. In *OSDI'00: Proceedings of the 4th conference on Symposium on Operating System Design & Implementation*, pages 6–6, Berkeley, CA, USA, 2000. USENIX Association.

[88] S. Guha, N. Daswani, and R. Jain. An experimental study of the Skype peer-to-peer VoIP system. In *Proceedings of IPTPS'06, Santa Barbara, CA*, February 2006.

[89] Ramakrishna Gummadi, Omprakash Gnawali, and Ramesh Govindan. Macro-programming wireless sensor networks using Kairos. In *Distributed Computing in Sensor Systems, First IEEE International Conference (DCOSS 2005), Marina del Rey, CA, USA*, volume 3560 of *Lecture Notes in Computer Science*, pages 126–140. Springer, June 2005.

[90] P. Gupta and P. R. Kumar. The capacity of wireless networks. *Information Theory, IEEE Transactions on*, 46(2):388–404, 2000.

[91] Thomas Hammel and Mark Rich. A higher capability sensor node platform suitable for demanding applications. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 138–147, New York, NY, USA, 2007. ACM Press.

[92] Chih-Chieh Han, Ram Kumar, Roy Shea, Eddie Kohler, and Mani Srivastava. A dynamic operating system for sensor nodes. In *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 163–176, New York, NY, USA, 2005. ACM Press.

[93] Steven Hand and Timothy Roscoe. Mnemosyne: Peer-to-Peer Steganographic Storage. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Boston, MA, USA*, March 2002.

[94] Matthew Harren, Joseph M. Hellerstein, Ryan Huebsch, Boon Thau Loo, Scott Shenker, and Ion Stoica. Complex queries in DHT-based peer-to-peer networks. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 242–259, London, UK, 2002. Springer-Verlag.

[95] N. J. Harvey, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman. Skipnet: A scalable overlay network with practical locality properties. In *Proceedings of USITS (Seattle, WA, March 2003), USENIX*, 2003.

[96] Les Hatton. *Safer C: Developing software for high-integrity and safety-critical systems.* McGraw-Hill, Maidenhead, 1994.

[97] T. He, J. Stankovic, C. Lu, and T. Abdelzaher. SPEED: A real-time routing protocol for sensor networks. Technical Report CS-2002-09, University of Virginia, March 2002.

[98] Tian He, Sudha Krishnamurthy, John A. Stankovic, Tarek Abdelzaher, Liqian Luo, Radu Stoleru, Ting Yan, Lin Gu, Jonathan Hui, and Bruce Krogh. Energy-efficient surveillance system using wireless sensor networks. In *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 270–283, New York, NY, USA, 2004. ACM Press.

[99] Tian He, John A. Stankovic, Chenyang Lu, and Tarek Abdelzaher. SPEED: A stateless protocol for real-time communication in sensor networks. In *ICDCS '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*, page 46, Washington, DC, USA, 2003. IEEE Computer Society.

[100] C. L. Hedrick. RFC 1058: Routing Information Protocol. Downloaded from http://www.ietf.org/rfc/rfc1058.txt (checked 28/05/2007), 1988.

[101] J. Heidemann, N. Bulusu, J. Elson, C. Intanagonwiwat, K. chan Lan, Y. Xu, W. Ye, D. Estrin, and R. Govindan. Effects of detail in wireless network simulation. In *Proceedings of the SCS Multiconference on Distributed Simulation, Phoenix, Arizona, USA*, pages 3–11. USC/Information Sciences Institute, Society for Computer Simulation, January 2001.

[102] John Heidemann, Fabio Silva, Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, and Deepak Ganesan. Building efficient wireless sensor networks with low-level naming. In *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 146–159, New York, NY, USA, 2001. ACM Press.

[103] Wendi Beth Heinzelman, Amy L. Murphy, Hervaldo S. Carvalho, and Mark A. Perillo. Middleware to support sensor network applications. *IEEE Network*, 18(1):6–14, 2004.

[104] A. Helmy. Small worlds in wireless networks. *IEEE Communications Letters*, 7(10):490–492, October 2003.

[105] Thomas R. Henderson, Sumit Roy, Sally Floyd, and George F. Riley. ns-3 project goals. In *WNS2 '06: Proceeding from the 2006 workshop on ns-2: the IP network simulator*, page 13, New York, NY, USA, 2006. ACM Press.

[106] Stephan Hengstler, Daniel Prashanth, Sufen Fong, and Hamid Aghajan. Mesheye: a hybrid-resolution smart camera mote for applications in distributed intelligent surveillance. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 360–369, New York, NY, USA, 2007. ACM Press.

[107] Karen Henricksen and Ricky Robinson. A survey of middleware for sensor networks: state-of-the-art and future directions. In *MidSens '06: Proceedings of the international workshop on Middleware for sensor networks*, pages 60–65, New York, NY, USA, 2006. ACM Press.

[108] Kirsten Hildrum, John D. Kubiatowicz, Satish Rao, and Ben Y. Zhao. Distributed data location in a dynamic network. Technical Report UCB/CSD-02-1178, University of California at Berkeley, April 2002. Updated version to appear in SPAA 2002.

[109] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David E. Culler, and Kristofer S. J. Pister. System architecture directions for networked sensors. In *Architectural Support for Programming Languages and Operating Systems*, pages 93–104, 2000.

[110] Chung-Hsing Hsu, Ulrich Kremer, and Michael Hsiao. Compiler-directed dynamic frequency and voltage scheduling. In *PACS '00: Proceedings of the First International Workshop on Power-Aware Computer Systems-Revised Papers*, pages 65–81, London, UK, 2001. Springer-Verlag.

[111] Lingxuan Hu and David Evans. Secure aggregation for wireless networks. In *SAINT-W '03: Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT'03 Workshops)*, page 384, Washington, DC, USA, 2003. IEEE Computer Society.

[112] Lingxuan Hu and David Evans. Localization for mobile sensor networks. In *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*, pages 45–57, New York, NY, USA, 2004. ACM Press.

[113] Polly Huang, Deborah Estrin, and John S. Heidemann. Enabling large-scale simulations: Selective abstraction approach to the study of multicast protocols. In *MASCOTS*, pages 241–248, 1998.

[114] Jean-Pierre Hubaux, Levente Buttyán, and Srdan Čapkun. The quest for security in mobile ad hoc networks. In *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 146–155, New York, NY, USA, 2001. ACM Press.

[115] D. A. Huffman. A method for the construction of minimum redundancy codes. In *Proc. IRE 40*, pages 1098–1101, 1952.

[116] Chalermek Intanagonwiwat, Deborah Estrin, Ramesh Govindan, and John Heidemann. Impact of network density on data aggregation in wireless sensor networks. In *ICDCS '02: Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*, page 457, Washington, DC, USA, 2002. IEEE Computer Society.

[117] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Mobile Computing and Networking*, pages 56–67, 2000.

[118] Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. Netw.*, 11(1):2–16, 2003.

[119] Ravindra Jejurikar and Rajesh Gupta. Energy aware task scheduling with task synchronization for embedded real time systems. In *CASES '02: Proceedings of the 2002 international conference on Compilers, architecture, and synthesis for embedded systems*, pages 164–169, New York, NY, USA, 2002. ACM Press.

[120] Wong K J Arvind D K. Specknets: New challenges for wireless communication protocols. In *Proceedings of The IEEE International Conference on Information Technology and Applications*, volume 2, pages 728–733, Australia, July 2005.

[121] J. Kahn, R. Katz, and K. Pister. Emerging challenges: Mobile networking for smart dust. *J. Comm. Networks*, pages 188–196, September 2000.

[122] Chris Karlof, Naveen Sastry, and David Wagner. TinySec: A link layer security architecture for wireless sensor networks. In *SenSys 04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 162–175, Baltimore, November 2004.

[123] Sukun Kim, Shamim Pakzad, David Culler, James Demmel, Gregory Fenves, Steven Glaser, and Martin Turon. Health monitoring of civil infrastructures using wireless sensor networks. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 254–263, New York, NY, USA, 2007. ACM Press.

[124] Jiejun Kong and Mario Gerla. Providing real-time security support for multi-level ad-hoc networks. In *IEEE Military Communications Conference (MILCOM'02), Anaheim, California, USA*, pages 1350–1355, October 2002.

[125] Yannis Kotidis. Snapshot queries: Towards data-centric sensor networks. In *ICDE '05: Proceedings of the 21st International Conference on Data Engineering (ICDE'05)*, pages 131–142, Washington, DC, USA, 2005. IEEE Computer Society.

[126] D. Kotz, C. Newport, and C. Elliott. The mistaken axioms of wireless-network research. Technical Report TR2003-467, Dept. of Computer Science, Dartmouth College, July 2003.

[127] R. Kumar, K. Farkas, N. Jouppi, P. Ranganathan, and D. Tullsen. A multi-core approach to addressing the energy-complexity problem in microprocessors. In *Proceedings of the Workshop on Complexity-Effective Design(WCED'03)*, June 2003.

[128] J. F. Kurose, M. Schwartz, and Y. Yemini. *Tutorial: hard real-time systems*, chapter Multiple-access protocols and time-constrained communication, pages 432–459. IEEE Computer Society Press, Los Alamitos, CA, USA, 1989.

[129] N. Kushalnagar, G. Montenegro, and C. Schumacher. 6LoWPAN: Overview, assumptions, problem statement and goals. http://www.ietf.org/internet-drafts/draft-ietf-6lowpan-problem-08.txt, February 2007. accessed 12/05/2007.

[130] O. Landsiedel, K. Wehrle, B. Titzer, and J. Palsberg. Enabling detailed modeling and analysis of sensor networks. *Praxis der Informationsverarbeitung und Kommunikation*, 2005.

[131] Olaf Landsiedel, Katharina Anna Lehmann, and Klaus Wehrle. T-DHT: Topology-based Distributed Hash Tables. In *P2P '05: Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing (P2P'05)*, pages 143–144, Washington, DC, USA, 2005. IEEE Computer Society.

[132] Loukas Lazos, Radha Poovendran, and James A. Ritcey. Probabilistic detection of mobile targets in heterogeneous sensor networks. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 519–528, New York, NY, USA, 2007. ACM Press.

[133] HyungJune Lee, Alberto Cerpa, and Philip Levis. Improving wireless simulation through noise modeling. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 21–30, New York, NY, USA, 2007. ACM Press.

[134] J. Lehnert, D. Gsrgen, H. Frey, and P. Sturm. A scalable workbench for implementing and evaluating distributed applications in mobile ad hoc networks. In *Proceedings of Mobile Ad Hoc Networks, Western Simulation MultiConference WMC'04, San Diego, California, USA*, 2004.

[135] Will E. Leland, Murad S. Taqqu, Walter Willinger, and Daniel V. Wilson. On the self-similar nature of Ethernet traffic. In *SIGCOMM '93: Conference proceedings on Communications architectures, protocols and applications*, pages 183–193, New York, NY, USA, 1993. ACM Press.

[136] P. Levis, D. Gay, and D. Culler. Active sensor networks. In *Proceedings of the 2nd USENIX/ACM Symposium on Network Systems Design and Implementation (NSDI), 2005*, May 2005.

[137] Philip Levis and David Culler. Mate: a tiny virtual machine for sensor networks. In *ASPLOS-X: Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, pages 85–95, New York, NY, USA, 2002. ACM Press.

[138] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. TOSSIM: accurate and scalable simulation of entire TinyOS applications. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 126–137, New York, NY, USA, 2003. ACM Press.

[139] H. Li, P. Shenoy, and K. Ramamritham. Scheduling communication in real-time sensor applications. In *RTAS '04: Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'04)*, page 10, Washington, DC, USA, 2004. IEEE Computer Society.

[140] Mo Li and Yunhao Liu. Underground structure monitoring with wireless sensor networks. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 69–78, New York, NY, USA, 2007. ACM Press.

[141] Jane Liu. *Real-Time Systems*. Prentice-Hall, New Jersey, 2000.

[142] Jane Liu, Kwei-Jay Lin, Riccardo Bettati, David Hull, and Albert Yu. *Paradigms for Dependable Applications*, chapter 3.1. Foundations of Dependable Computing. Kluwer, 1st edition, September 1994.

[143] Jane W. S. Liu, Kwei-Jay Lin, Wei-Kuan Shih, Albert Chuang shi Yu, Jen-Yao Chung, and Wei Zhao. Algorithms for scheduling imprecise computations. *Computer*, 24(5):58–68, 1991.

[144] Jane W. S. Liu and Wei-Kuan Shih. Algorithms for scheduling imprecise computations with timing constraints to minimize maximum error. *IEEE Trans. Comput.*, 44(3):466–471, 1995.

[145] Ke Liu, Nael Abu-Ghazaleh, and Kyoung-Don Kang. JiTS: Just-in-time scheduling for real-time sensor data dissemination. In *PERCOM '06: Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications (PERCOM'06)*, pages 42–46, Washington, DC, USA, 2006. IEEE Computer Society.

[146] Ting Liu and Margaret Martonosi. Impala: a middleware system for managing autonomic, parallel sensor systems. In *PPoPP '03: Proceedings of the ninth ACM SIGPLAN symposium on Principles and practice of parallel programming*, pages 107–118, New York, NY, USA, 2003. ACM Press.

[147] Xiaomei Liu. Location awareness in unstructured peer-to-peer systems. *IEEE Trans. Parallel Distrib. Syst.*, 16(2):163–174, 2005. Member-Yunhao Liu and Member-Li Xiao and Fellow-Lionel M. Ni and Senior Member-Xiaodong Zhang.

[148] Jacob R. Lorch and Alan Jay Smith. Scheduling techniques for reducing processor energy use in macos. *Wirel. Netw.*, 3(5):311–324, 1997.

[149] Chenyang Lu, Brian M. Blum, Tarek F. Abdelzaher, John A. Stankovic, and Tian He. RAP: A real-time communication architecture for large-scale wireless sensor networks. In *RTAS '02: Proceedings of the Eighth IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'02)*, page 55, Washington, DC, USA, 2002. IEEE Computer Society.

[150] Chenyang Lu, John A. Stankovic, Sang H. Son, and Gang Tao. Feedback control real-time scheduling: Framework, modeling, and algorithms*. *Real-Time Syst.*, 23(1-2):85–126, 2002.

[151] Mark Luk, Ghita Mezzour, Adrian Perrig, and Virgil Gligor. MiniSec: a secure sensor network communication architecture. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 479–488, New York, NY, USA, 2007. ACM Press.

[152] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In *SIGMETRICS '02: Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 258–259, New York, NY, USA, 2002. ACM Press.

[153] Qin Lv, Sylvia Ratnasamy, and Scott Shenker. Can heterogeneity make Gnutella scalable? In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 94–103, London, UK, 2002. Springer-Verlag.

[154] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. TAG: a tiny aggregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):131–146, 2002.

[155] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. The design of an acquisitional query processor for sensor networks. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 491–502, New York, NY, USA, 2003. ACM Press.

[156] Samuel Madden, Robert Szewczyk, Michael J. Franklin, and David Culler. Supporting aggregate queries over ad-hoc wireless sensor networks. In *WMCSA '02: Proceedings of*

the Fourth IEEE Workshop on Mobile Computing Systems and Applications, page 49, Washington, DC, USA, 2002. IEEE Computer Society.

[157] Sam Malek and Marija Mikic-Rakic. A style-aware architectural middleware for resource-constrained, distributed systems. *IEEE Trans. Softw. Eng.*, 31(3):256–272, 2005. Member-Nenad Medvidovic.

[158] Evangelos P. Markatos. Tracing a large-scale peer to peer system: an hour in the life of Gnutella. In *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2002.

[159] Pedro José Marrón, Andreas Lachenmann, Daniel Minder, Jörg Hähner, Robert Sauter, and Kurt Rothermel. TinyCubus: A flexible and adaptive framework for sensor networks. In *Proceedings of the Second European Workshop on Wireless Sensor Networks (EWSN 2005)*, pages 278–289, January 2005.

[160] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Mobile Computing and Networking*, pages 255–265, 2000.

[161] W. Matthew, J. Miller, and N. Vaidya. A hybrid network implementation to extend infrastructure reach. Technical report, University of Illinois at Urbana-Champaign, January 2003.

[162] Thomas Maufer. *IP Fundamentals.* Prentice-Hall, 1999.

[163] Terry May, Shaun Dunning, and Jason Hallstrom. An RPC design for wireless sensor networks. In *Second IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS05), Washington DC, USA*, November 2005.

[164] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *Proceedings of International Workshop on Peer-to-Peer Systems (IPTPS'02), Cambridge, MA*, 2002.

[165] William P. McCartney and Nigamanth Sridhar. TOSDev: a rapid development environment for tinyos. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 387–388, New York, NY, USA, 2006. ACM Press.

[166] John McQuillan and Ira Richer. *Computer Networks and Simulation*, chapter A new network simulation technique, pages 187–193. North Holland, Amsterdam, 1st edition, 1978.

[167] Vivek Mhatre and Catherine Rosenberg. Design guidelines for wireless sensor networks: communication, clustering and aggregation. *Ad Hoc Networks*, 2(1):45–63, 2004.

[168] S. Milgram. The small world problem. *Psychology Today*, 2(1):60–67, 1967.

[169] D. Musiani, K. Lin, and T. Simunic Rosing. Active sensing platform for wireless structural health monitoring. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 390–399, New York, NY, USA, 2007. ACM Press.

[170] Sule Nair and Rachel Cardell-Oliver. Formal specification and analysis of performance variation in sensor network diffusion protocols. In *MSWiM '04: Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 170–173, New York, NY, USA, 2004. ACM Press.

[171] Valeri Naoumov and Thomas Gross. Simulation of large ad hoc networks. In *MSWIM '03: Proceedings of the 6th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems*, pages 50–57, New York, NY, USA, 2003. ACM Press.

[172] James Newsome, Elaine Shi, Dawn Song, and Adrian Perrig. The sybil attack in sensor networks: analysis & defenses. In *IPSN '04: Proceedings of the third international symposium on Information processing in sensor networks*, pages 259–268, New York, NY, USA, 2004. ACM Press.

[173] Ryan Newton, Greg Morrisett, and Matt Welsh. The regiment macroprogramming system. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 489–498, New York, NY, USA, 2007. ACM Press.

[174] Gultekin Ozsoyoglu and Richard T. Snodgrass. Temporal and real-time databases: A survey. *Knowledge and Data Engineering*, 7(4):513–532, 1995.

[175] M. Tamer Özsu and Patrick Valduriez. Distributed and parallel database systems. *ACM Computing Surveys*, 28(1):125–128, 1996.

[176] P. Papadimitratos and Z. Haas. Secure routing for mobile ad hoc networks. In *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002), San Antonio, TX*, January 2002.

[177] Heemin Park, Jeff Burke, and Mani B. Srivastava. Design and implementation of a wireless sensor network for intelligent light control. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 370–379, New York, NY, USA, 2007. ACM Press.

[178] Bryan Parno, Adrian Perrig, and Virgil Gligor. Distributed detection of node replication attacks in sensor networks. In *SP '05: Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 49–63, Washington, DC, USA, 2005. IEEE Computer Society.

[179] M. Paulk, editor. *The Capability Maturity Model: Guidelines for improving the software process.* Addison-Wesley, Cambridge, MA, 1996.

[180] Adrian Perrig, Robert Szewczyk, Victor Wen, David E. Culler, and J. D. Tygar. SPINS: security protocols for sensor netowrks. In *Mobile Computing and Networking*, pages 189–199, 2001.

[181] L. F. Perrone and D. M. Nicol. A scalable simulator for TinyOS applications. In *WSC '02: Proceedings of the 2002 Winter Simulation Conference (WSC'02) - Volume 1*, pages 679–687, Washington, DC, USA, 2002. IEEE Computer Society.

[182] Larry L. Peterson. The profile naming service. *ACM Trans. Comput. Syst.*, 6(4):341–364, 1988.

[183] Padmanabhan Pillai and Kang G. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. In *ACM Symposium on Operating Systems Principles*, pages 89–102, 2001.

[184] C. Greg Plaxton, Rajmohan Rajaraman, and Andrea W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *ACM Symposium on Parallel Algorithms and Architectures*, pages 311–320, 1997.

[185] Marius Portmann, Sebastien Ardon, Patrick Senac, and Aruna Seneviratne. PROST: A programmable structured peer-to-peer overlay network. In *P2P '04: Proceedings of the Fourth International Conference on Peer-to-Peer Computing (P2P'04)*, pages 280–281, Washington, DC, USA, 2004. IEEE Computer Society.

[186] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Commun. ACM*, 43(5):51–58, 2000.

[187] Simon Poulding, Paul Emberson, Iain Bate, and John Clark. An efficient experimental methodology for configuring search-based design algorithms. Technical report, University of York, Computer Science Department, 2007.

[188] D. Prasad, A. Burns, and Atkin M. The measurement and usage of utility in adaptive real-time systems. *Journal of Real-Time Systems*, 25(2/3):277–296, 2003.

[189] Himabindu Pucha, Saumitra M. Das, and Y. Charlie Hu. Ekta: An efficient DHT substrate for distributed applications in mobile ad hoc networks. In *WMCSA '04: Proceedings of the Sixth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'04)*, pages 163–173, Washington, DC, USA, 2004. IEEE Computer Society.

[190] V. Raghunathan, C. Schurgers, S. Park, and M. Srivastava. Energy aware wireless microsensor networks. *IEEE Signal Processing Magazine*, 19(2):40–50, March 2002.

[191] Krithi Ramamritham. Real-time databases. *Distributed and Parallel Databases*, 1(2):199–226, 1993.

[192] D. Rao and P. Wilsey. Multi-resolution network simulations using dynamic component substitution. In *MASCOTS '01: Proceedings of the Ninth International Symposium in Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'01)*, page 142, Washington, DC, USA, 2001. IEEE Computer Society.

[193] Dhananjai M. Rao and Philip A. Wilsey. Modeling and simulation of active networks. In *SS '01: Proceedings of the 34th Annual Simulation Symposium (SS01)*, page 177, Washington, DC, USA, 2001. IEEE Computer Society.

[194] Dhananjai Madhava Rao and Philip A. Wilsey. Simulation of ultra-large communication networks. In *MASCOTS*, pages 112–119, 1999.

[195] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu. Data-centric storage in sensornets with GHT, a geographic hash table. *Mob. Netw. Appl.*, 8(4):427–442, 2003.

[196] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content addressable network. Technical Report TR-00-010, University of California at Berkeley, Berkeley, CA, 2000.

[197] Sylvia Ratnasamy, Mark Handley, Richard M. Karp, and Scott Shenker. Topologically-aware overlay construction and server selection. In *INFOCOM*, 2002.

[198] Shansi Ren, Lei Guo, Song Jiang, and Xiaodong Zhang. SAT-Match: A self-adaptive topology matching method to achieve low lookup latency in structured P2P overlay networks. In *IPDPS*, 2004.

[199] George F. Riley, Richard Fujimoto, and Mostafa H. Ammar. A generic framework for parallelization of network simulations. In *MASCOTS*, pages 128–, 1999.

[200] Kay Römer. Programming paradigms and middleware for sensor networks. *GI/ITG Workshop on Sensor Networks*, pages 49–54, 2004.

[201] Kay Römer, Oliver Kasten, and Friedemann Mattern. Middleware challenges for wireless sensor networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(4):59–61, 2002.

[202] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329+, 2001.

[203] Masoomeh Rudafshani and Suprakash Datta. Localization in wireless sensor networks. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 51–60, New York, NY, USA, 2007. ACM Press.

[204] Basir Sakandar and David Barnes. *Cisco LAN Switching Fundamentals.* Cisco Press Fundamentals Series. Cisco Press, 2nd edition, September 2004.

[205] S. Saroiu, P. Gummadi, and S. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networking (MMCN) 2002*, January 2002.

[206] Anna Scaglione and Sergio D. Servetto. On the interdependence of routing and data compression in multi-hop sensor networks. In *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 140–147, New York, NY, USA, 2002. ACM Press.

[207] Douglas C. Schmidt. Middleware for real-time and embedded systems. *Commun. ACM*, 45(6):43–48, 2002.

[208] Bruce Schneier. *Secrets and Lies: Digital security in a networked world.* Wiley, New York, first edition, 2000.

[209] Karim Seada and Ahmed Helmy. Rendezvous regions: A scalable architecture for service location and data-centric storage in large-scale wireless networks. *ipdps*, 13:218a, 2004.

[210] Scott Shenker, Sylvia Ratnasamy, Brad Karp, Ramesh Govindan, and Deborah Estrin. Data-centric storage in sensornets. *SIGCOMM Comput. Commun. Rev.*, 33(1):137–142, 2003.

[211] M. Shepperd and D. Ince. *Derivation and validation of software metrics.* Clarendon Press, Oxford, 1993.

[212] Abraham Silberschatz, Peter Galvin, and James L. Peterson. *Operating System Concepts.* Wiley, New York, fifth edition, February 1998.

[213] F. Silva, J. Heidemann, R. Govindan, and D. Estrin. Directed diffusion. Technical Report ISI-TR-2004-586, USC/Information Sciences Institute, January 2004.

[214] B. Silverman. Wavelets in statistics: beyond the standard assumptions. *Phil. Trans. Roy. Soc. Lond.*, (357), 1999.

[215] Jaspreet Singh, Upamanyu Madhow, Rajesh Kumar, Subhash Suri, and Richard Cagley. Tracking multiple targets using binary proximity sensors. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 529–538, New York, NY, USA, 2007. ACM Press.

[216] Stephen So, Farinaz Koushanfar, Anatoliy Kosterev, and Frank Tittel. LaserSPECks: laser spectroscopic trace-gas sensor networks - sensor integration and applications. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 226–235, New York, NY, USA, 2007. ACM Press.

[217] A. Sobeih, M. Viswanathan, and J. Hou. Check and simulate: A case for incorporating model checking in network simulation. In *Proc. of ACM-IEEE MEMOCODE'04*, June 2004.

[218] P. Sousa, P. Carvalho, and V. Freitas. Scheduling time-sensitive IP traffic. In G. Goos et al, editor, *Proc. of 6th IFIP/IEEE MMNS International Conference, Belfast, Northern Ireland*, pages 368–380. Springer-Verlag, September 2003.

[219] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis. Some implications of low power wireless to IP networking. In *Proceedings of HotOS V*, November 2006.

[220] Frank Stajano and Ross J. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Proceedings of the 7th International Workshop on Security Protocols*, pages 172–194, London, UK, 2000. Springer-Verlag.

[221] J. A. Stankovic, T. E. Abdelzaher, C. Lu, L. Sha, and J. C. Hou. Real-time communication and coordination in embedded sensor networks. *Proc. IEEE*, 91(7):1002–1022, 2003.

[222] John Stankovic. DARPA ClearPoint presentation slides - VigilNet. http://www.cs.virginia.edu/∼stankovic/psfiles/cleanpoint-talk-dec04.ppt (checked 22/06/2007), December 2004.

[223] John A. Stankovic. Distributed computing. Technical Report UM-CS-1992-011, University of Massachusetts at Amherst, 1992.

[224] John A. Stankovic, Tian He, Tarek Abdelzaher, Mike Marley, Gang Tao, Sang Son, and Cenyan Lu. Feedback control scheduling in distributed real-time systems. In *RTSS '01: Proceedings of the 22nd IEEE Real-Time Systems Symposium (RTSS'01)*, page 59, Washington, DC, USA, 2001. IEEE Computer Society.

[225] Peter Stavroulakis, editor. *Chaos Applications in Telecommunications.* CRC Press, Boca Raton, FL, 1st edition, 2006.

[226] Ivan Stoianov, Lama Nachman, Sam Madden, and Timur Tokmouline. PIPENET - a wireless sensor network for pipeline monitoring. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 264–273, New York, NY, USA, 2007. ACM Press.

[227] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.

[228] Sameer Sundresh, Wooyoung Kim, and Gul Agha. SENS: A sensor, environment and network simulator. In *37th Annual Simulation Symposium (ANSS37)*, 2004.

[229] Vishu Swaminathan and Krishnendu Chakrabarty. Investigating the effect of voltage-switching on low-energy task scheduling in hard real-time systems. In *ASP-DAC '01: Proceedings of the 2001 conference on Asia South Pacific design automation*, page 251, New York, NY, USA, 2001. ACM Press.

[230] W. Sweldens and P. Schröder. Building your own wavelets at home. In *Wavelets in Computer Graphics*, pages 15–87. ACM SIGGRAPH Course notes, 1996.

[231] Robert Szewczyk, Alan Mainwaring, Joseph Polastre, John Anderson, and David Culler. An analysis of a large scale habitat monitoring application. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 214–226, New York, NY, USA, 2004. ACM Press.

[232] Robert Szewczyk, Joseph Polastre, Alan Mainwaring, and David Culler. Lessons from a sensor network expedition. In *Proceedings of the First European Workshop on Sensor Networks (EWSN)*, January 2004.

[233] Hongsuda Tangmunarunkit, Ramesh Govindan, Sugih Jamin, Scott Shenker, and Walter Willinger. Network topologies, power laws, and hierarchy. *SIGCOMM Comput. Commun. Rev.*, 32(1):76–76, 2002.

[234] M. Taqqu and V. Teverosky. Is network traffic self-similar or multifractal? *Fractals*, 5(1):63–73, 1997.

[235] Jonathan Tate. Qualifying Dissertation: Wireless Sensor Networks (full version). http://www.cs.york.ac.uk/∼jt/docs/qdfull.pdf, June 2007.

[236] David L. Tennenhouse, Jonathan M. Smith, W. David Sincoskie, David J. Wetherall, and Gary J. Minden. A survey of active network research. *IEEE Communications Magazine*, 35(1):80–86, 1997.

[237] S. Tilak, V. Kolar, N. B. Abu-Ghazaleh, and K. D. Kang. Dynamic localization control for mobile sensor networks. In *IEEE International Workshop on Strategies for Energy Efficiency in Ad Hoc and Sensor Networks (IEEE IWSEEASN'05)*, April 2005.

[238] Sameer Tilak, Nael B. Abu-Ghazaleh, and Wendi Heinzelman. A taxonomy of wireless micro-sensor network models. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(2):28–36, 2002.

[239] Ben L. Titzer, Daniel K. Lee, and Jens Palsberg. Avrora: scalable sensor network simulation with precise timing. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, page 67, Piscataway, NJ, USA, 2005. IEEE Press.

[240] Chai-Keong Toh, Petri Mahonen, and Mikko Uusitalo. Standardization efforts and future research issues for wireless sensors and mobile ad hoc networks. *IEICE Trans. Commun.*, E88-B(9):3500–3507, September 2005.

[241] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostic, J. Chase, and D. Becker. Scalability and accuracy in a large-scale network emulator. In *Proc. 5th OSDI*, December 2002.

[242] Nitin H. Vaidya, Paramvir Bahl, and Seema Gupta. Distributed fair scheduling in a wireless LAN. In *Mobile Computing and Networking*, pages 167–178, 2000.

[243] Darryl Veitch, Nicolas Hohn, and Patrice Abry. Multifractality in TCP/IP traffic: the case against. *Comput. Networks*, 48(3):293–313, 2005.

[244] David Wagner. Resilient aggregation in sensor networks. In *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 78–87, New York, NY, USA, 2004. ACM Press.

[245] Tim Wark, Chris Crossman, Wen Hu, Ying Guo, Philip Valencia, Pavan Sikka, Peter Corke, Caroline Lee, John Henshall, Kishore Prayaga, Julian O'Grady, Matt Reed, and Andrew Fisher. The design and evaluation of a mobile sensor/actuator network for

autonomous animal control. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 206–215, New York, NY, USA, 2007. ACM Press.

[246] Brett Warneke, Matt Last, Brian Liebowitz, and Kristofer S. J. Pister. Smart dust: Communicating with a cubic-millimeter computer. *Computer*, 34(1):44–51, 2001.

[247] Mark Weiser, Brent Welch, Alan J. Demers, and Scott Shenker. Scheduling for reduced CPU energy. In *Operating Systems Design and Implementation*, pages 13–23, 1994.

[248] M. Welsh and G. Mainland. Programming sensor networks using abstract regions. In *Proc. First Symp. Networked Systems Design and Implementation (NSDI'04), New York*, pages 29–42. ACM Press, March 2004.

[249] Brian White, Jay Lepreau, and Shashi Guruprasad. Lowering the barrier to wireless and mobile experimentation. *SIGCOMM Comput. Commun. Rev.*, 33(1):47–52, 2003.

[250] Ian H. Witten, Radford M. Neal, and John G. Cleary. Arithmetic coding for data compression. *Commun. ACM*, 30(6):520–540, 1987.

[251] M. Wolenetz, R. Kumar, J. Shin, and U. Ramachandran. Middleware guidelines for future sensor networks. In *Proceedings of the First Workshop on Broadband Advanced Sensor Networks, San Jose, California, USA*, October 2004.

[252] J. Liu Y. Chen, A. Liestman. A hierarchical energy-efficient framework for data aggregation in wireless sensor networks. *IEEE Transactions on Vehicular Technology*, 2005.

[253] Yi Yang, Xinran Wang, Sencun Zhu, and Guohong Cao. SDAP: a secure hop-by-hop data aggregation protocol for sensor networks. In *MobiHoc '06: Proceedings of the seventh ACM international symposium on Mobile ad hoc networking and computing*, pages 356–367, New York, NY, USA, 2006. ACM Press.

[254] W. Yeong, T. Howes, and S. Kille. RFC 1777: Lightweight Directory Access Protocol. Downloaded from http://www.ietf.org/rfc/rfc1777.txt (checked 28/05/2007), 1995.

[255] B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, Computer Science Division, U. C. Berkeley, April 2001.

[256] Ben Y. Zhao, Yitao Duan, Ling Huang, Anthony D. Joseph, and John Kubiatowicz. Brocade: Landmark routing on overlay networks. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 34–44, London, UK, 2002. Springer-Verlag.

[257] Lidong Zhou and Zygmunt J. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, 1999.

[258] Bo Zhu, Zhiguo Wan, Mohan S. Kankanhalli, Feng Bao, and Robert H. Deng. Anonymous secure routing in mobile ad-hoc networks. In *LCN '04: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN'04)*, pages 102–108, Washington, DC, USA, 2004. IEEE Computer Society.

[259] Shelley Q. Zhuang, Ben Y. Zhao, Anthony D. Joseph, Randy H. Katz, and John D. Kubiatowicz. Bayeux: an architecture for scalable and fault-tolerant wide-area data dissemination. In *NOSSDAV '01: Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video*, pages 11–20, New York, NY, USA, 2001. ACM Press.

[260] Hubert Zimmerman. OSI reference model - the ISO model of architecture for open systems interconnection. *IEEE Transactions on Communications*, 28(4):425–432, 1980.

[261] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.