Load Sharing and Bandwidth Control in Mobile P2P Wireless Sensor Networks

Elisa Rondini and Stephen Hailes
Dept. of Computer Science
University College London
Gower Street, London WC1E 6BT, UK
{E.Rondini | S.Hailes}@cs.ucl.ac.uk

Li Li
Communications Research Centre Canada
Ottawa, Canada
li.li@crc.ca

Abstract

In this paper, we investigate the problem of resource constraints in Mobile Peer-to-Peer Wireless Sensor Networks (MP2P WSNs). We propose a scheme to load share tasks among peer sensor nodes, taking into account both their computational capabilities and networking conditions. Experiments to evaluate the performance results were conducted over real Tmote Sky sensor testbeds. We demonstrate that significant performance improvements in terms of latency can be achieved for a MP2P WSN by considering both load and network constraints, and we argue that, when ignoring the latter, the performance of MP2P computing in the application overlay could be severely impacted.

1 Introduction

Modern Wireless Sensor Networks (WSNs) have the potential to provide situational information to enhance the effectiveness of mission critical operations. However, sensors often generate large amounts of data at a (fine) granularity that is not operationally useful. Peer-to-Peer (P2P) overlays can thus be employed in WSNs to enable effective data query and search based on attributes [3]. For example, to monitor fire hazards, the application needs to identify sensor nodes at which temperature, smoke and visible light readings have passed a given threshold. It must then execute a fusion process to estimate the location, importance and other related attributes of the situation. Such applications involve query and search on attribute identifiers instead of known IP addresses, and require collaborative task execution among sensor peers based on the data attributes [17].

In comparison to general P2P networks, P2P WSNs give rise to new challenges as a result of bandwidth limitations in the underlying physical network: (i) Given the innate inaccuracy of such devices, data from multiple sensor nodes must normally be fused together to provide derived values in which one has confidence. (ii) Sensor nodes are much unusually heavily resource constrained, in terms of their computational strength, link capacity and battery power. For example, wireless networking protocols for sensors (e.g. 802.15.4) are severely limited in bandwidth (250 Kbps) compared to, say, WiFi (2–270 Mbps). Thus, to support an effective P2P application overlay on a WSN, resource management becomes critical in preventing the congestive collapse of the underlying physical network.

Various resource management schemes have been developed to distribute task execution (e.g. Grid Computing). However, such approaches usually assume high-end processors and connections that do not experience the bandwidth restrictions and congestion inevitable in WSNs. In recent years, first steps have been taken towards collaborative execution of applications in WSNs [19, 20, 4]. However, most such approaches focus on the CPU availability of the nodes during the task distribution process; they ignore network effects during and after distribution.

In this paper, we present our initial investigations of a novel peer task distribution scheme for Mobile P2P (MP2P) WSNs, which takes into account *both* the computational capabilities and the local network conditions of the peer sensor nodes. In particular, we present results from our implementation of two different task distribution algorithms on real Tmote Sky sensor testbeds. Our experimental results demonstrate that significant degradation in job execution time is experienced when ignoring the bandwidth availability in physical links between peers. We examine approaches to ameliorate this problem, and then discuss techniques by which the application overlay can infer physical route characteristics to guide the task distribution process.

The rest of the paper is organised as follows. Section 2 describes the MP2P WSN scenario. Section 3 discusses the MP2P load distribution problem analysing, with real experiments, the effects on the latency if local network conditions are ignored. Section 4 presents the performance results improved by the application of our novel task distribution scheme. Section 5 provides a brief overview of related work and, finally, Section 6 concludes the paper.

2 MP2P WSN Scenario Description

The network scenario of an example MP2P WSN is illustrated in Figure 1. A large number of sensor nodes are distributed throughout a given space, some static and predeployed, some mobile. Given the trade-off between computational power, battery capacity and form factor, nodes may be unable individually to compute tasks allocated to them. Neither can they simply ship all data back to a more capable sink because widespread contention for the medium might lead to network collapse. Thus, distributed computation is a key component in forming a robust and flexible sensor network system.

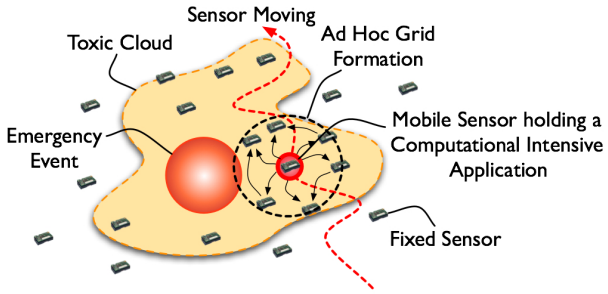

Figure 1. Example of a MP2P WSN.

When a job either requires data from multiple nodes, or requires computational power beyond that of a single node, peer nodes must be involved in the execution. The communication in organising this occupies bandwidth, but the result then requires less bandwidth to transmit throughout the wider network. As a prerequisite to the realisation of this MP2P computing paradigm in WSNs, it is necessary to establish whether, and to what extent, bandwidth considerations affect the dynamic task allocation process. It should be noted that the application overlay may have no, or very little knowledge of the underlying physical network. Moreover, node mobility adds to the system dynamism.

As an example, assume that an application needs to search in the west section of a tunnel for any location that has its temperature, smoke and light reading in excess of certain thresholds. Different readings need to be fused to determine if a fire is suspected at the location. For example, when several nodes report higher temperature readings within one area, a subsequent smoke reading must be gathered from one or several of the sensors in the vicinity. The data need to be fused with the temperature and possibly other readings to derive the current situation and estimated fire location and intensity. The application overlay issues the request indicating the data attributes that are of interest and this is routed by the MP2P network to reach the candidate peer nodes. The peers must then distribute the data processing task to the nodes that are best able to compute the required information in the least possible time. The first issue of searching and contacting the required peers was addressed in several other works [17, 3, 14]. In this work, we focus on effectively selecting peers from the candidate peer group to perform task execution with minimised job execution latency.

In a simple case, assume a node that needs to distribute a task, and two peer nodes to which it could potentially distribute it (Figure 2). One peer is rather less loaded than the other, but the most loaded of the peers is positioned in an environment with much lower network utilisation, making the choice difficult since the job execution latency is affected by both factors. Moreover, a distributed task might itself add to the congestion of an area in several ways: through the traffic resulting from the initial distribution of code and data, through subsequent communication between peers during the execution, and through control overhead. Thus, if we distribute a large task with substantial I/O requirements into an area of existing congestion, we will not only cause that task to run more slowly, but would also expect other tasks either within the area, or using the area for communication, to run more slowly too. Achieving the right balance is particularly critical in emergencies, since nodes (particularly those close to the incident) report data in large volumes. Injection of additional traffic in the affected area may lead to the failure of the network exactly when it is most needed.

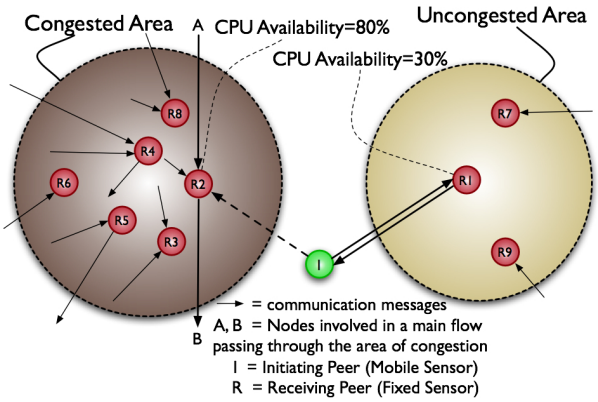

Figure 2. The Bandwidth Problem.

3 Problem of MP2P Load Distribution

The issue of timely medium access is one that originates somewhat below the MP2P application overlay. Indeed, MP2P application programmers are encouraged and enabled to ignore such issues. In the following sections, we demonstrate three things: (i) that congestion of links in the underlying network has a significant impact on task execution performance (Sec. 3.3); (ii) that it is nevertheless pos-

sible to ameliorate those effects using simple load sharing techniques if one has even rudimentary knowledge about the state of the network (Sec. 4.1); and (iii) that in MP2P systems one should adopt either a cross-layered approach or one based on heuristics at the application overlay to gather the parameters of the network status and apply them to the load sharing algorithms (Sec. 4.2). Before starting the analysis, we briefly describe our experimental setup (Sec. 3.1) and the load sharing algorithms we employed and adapted for our experiments (Sec. 3.2).

3.1 Experimental Setup

Our experimental testbed consists of Tmote Sky sensors running the Contiki OS [6]. This approach avoids the strongly simplifying assumptions that most simulators make about radio communications: they are often assumed to be error-free, with a circular transmission radius, with bidirectional communication, etc. In reality, such assumptions are so unreasonable that the validity of the work is questionable for even a small number of nodes [13]. For our experiments, we used the Heterogeneous Experimental Network (HEN) [1] deployed at the Department of Computer Science at UCL. We used actual computation, actual profiling of the medium, and actual network traffic.

During MP2P collaborations, *Initiating Peers* (IntPs) split the job into several tasks and then distribute them to *Receiving Peers* (RcvPs), for execution. Three phases of communication during task distribution are included in our experiments: (i) offload of the tasks from the IntP to the RcvP; (ii) internal communication exchanges needed to progress with task execution; (iii) communication upload of the combined final result from the RcvP to the IntP. In our tests, we used 3 IntP nodes, 21 RcvP nodes and a single streaming node whose sole task is to create network contention. We set the streaming sensor radio power level to provide a physical packet reception range of $\sim$250cm to congest one part of the involved area. Every job to be performed is split into at most 32 tasks.

Latency is used as our performance metric in all experiments because we assume that timeliness of information is vital in an emergency scenario. Moreover, latency captures the effects that tasks from one node have on the execution patterns of others. Results are obtained by computing the mean of the times measured by 3 lab-rat IntP nodes (each performing 50 different experimental runs). Job execution time is deemed to be the overall time spent to execute all the tasks into which a job is split. We examine three cases: (i) without network congestion, (ii) with congestion (extra traffic generated by the streaming node) but considering only computational requirements during load distribution, and (iii) with congestion and using our approach which additionally considers bandwidth requirements.

3.2 Load Sharing Algorithms

For task distribution, we employ two simple load sharing techniques: one reactive and sender-initiated (the Auction algorithm) and the other proactive and receiver-initiated (the Lookup List algorithm).

Auction Algorithm: In this, state information exchange is handled reactively. Each job is split into tasks; for each of these, the IntP broadcasts a task request message containing the details of the data type (attributes that the peer has to match), the task CPU and estimated bandwidth requirements. Upon receiving the task request, each RcvP that finds itself meeting the task requirements sends a bid to the IntP containing its CPU and bandwidth details. Once the IntP has received all bids, it chooses the best RcvP. Then it starts the offload of the task to the winner that consequently launches a process to compute the task, and sends the result back to the IntP. Multiple requests are handled by RcvPs on a First-Come First-Served (FCFS) basis.

Lookup List Algorithm: This is a more sophisticated algorithm, in which state information exchange is proactive. The main idea behind this algorithm was taken from [5] and adapted to allow for bandwidth control. At the beginning of the computation, every IntP launches a discovery phase in which a lookup list is filled with CPU and bandwidth availability details of all the neighbouring RcvPs. These lists are then dynamically updated during the execution of the algorithm. When an IntP has a task to distribute, it selects the best candidate RcvP from its lookup list and sends it a task request message containing the data type (attributes that the peer has to match), task CPU and bandwidth details. The RcvP sends an acknowledgement back to the IntP together with an update of its CPU and bandwidth availability. If the RcvP is able to execute the task, it indicates in the message its availability and it launches a process to receive the offload of the task, to execute the computation and then to send the result back together with another update of its resourse details. If the RcvP cannot execute the task, the IntP updates the CPU and bandwidth availability details of the previously chosen RcvP and selects the next best RcvP in its list. Multiple requests are managed by RcvPs using a FCFS strategy.

In both algorithms described above, there is a phase in which the IntP selects the peer node with the best resource availability to distribute the task execution. In Eq. 1, given an IntP, for each one of its RcvP neighbours i a weighted score function $S(i)$ is calculated to perform peer selection. $C(i)$ and $B(i)$ are defined as the CPU and bandwidth availability of i, and w_C and w_B the weights associated to them respectively. The RcvP with the highest score is selected for task distribution.

$$S(i) \quad = \quad w_C * C(i) + w_B * B(i) \quad i = 1, ..., N \qquad (1)$$

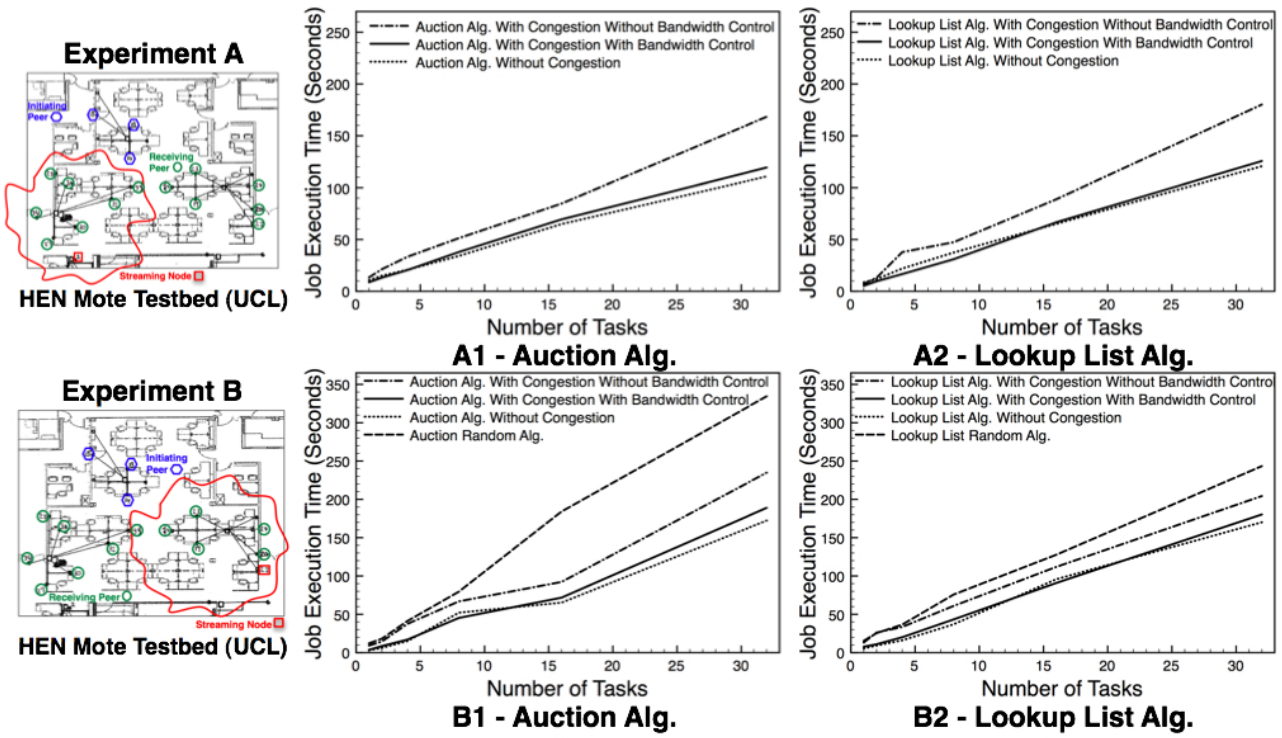

Figure 3. Experiment A and B nodes topology and results obtained on the HEN sensor testbed.

3.3 Effects of Congestion

In the first experiment, we evaluate the impact of network congestion during task distribution. To do that, we use the two load sharing algorithms described above, but only computing the CPU availability requirement $C(i)$ in the scoring function (i.e. $w_C \neq 0$ in Eq. 1). The bandwidth resource constraint is thus ignored (i.e. $w_B = 0$ in Eq. 1), an approach which is common to most of the existing load distribution algorithms. Note that $C(i)$ represents the number of processes that are actively running on a node.

In Figure 3, the results show that the addition of network contention leads to an increase in the job execution latency, with respect to the case without congestion, of ∼55% and ∼40% (Experiments A1 and B1) in the case of the Auction algorithm and ∼50% and ∼25% (Experiments A2 and B2) in the case of the Lookup List algorithm. This occurs when considering the simplest form of task offload to physically adjacent neighbours, and represents an upper bound on performance for MP2P WSNs, where similar problems of contention will be faced for each hop in a multihop route. Therefore, from this experiment we observe that the MP2P computing may encounter severe performance degradations if bandwidth requirements are not taken into account during task distribution.

4 Improving the Performance

4.1 Effects of Considering Bandwidth

We now explore whether taking into account bandwidth availability has a significant impact on job latency. The experiments are performed with the same configuration as above but feeding the algorithms, described in Sec. 3.2, with network information as well as computational load (i.e. $w_B \neq 0$ and $w_C \neq 0$ in Eq. 1). To gather network information, we elected to read the Clear Channel Assessment value from pin 28 of the Chipcon CC2420 transceiver and maintain a sliding window containing information for the last *n* temporal slots in which the radio channel was clear or busy. When aggregated, this value is used as an estimate of likely contention. We used the results of a number of runs to hand-tune the weights applied to computational load, w_C, and bandwidth availability, w_B, to 5 and 1 respectively.

The experiments demonstrate that by taking bandwidth into consideration, system performance approximates that of the uncongested scenario. In Figure 3 for both Experiment A and B, it can also be seen that the Auction and the Lookup List algorithms are comparable. When the number of tasks involved in each job is small, Lookup List may perform slightly better than Auction. When the numbers of IntP nodes and tasks are increased, Auction outper-

forms Lookup List. Note that the Auction algorithm fits better with the processing order of a conventional P2P system, where the search and query takes place first before the downloading starts. Although Auction appears to be heavier than Lookup List because of the initial search, it achieves better performance when the number of tasks increases, because each RcvP is always aware of the status of each IntP and vice versa when starting a task computation. Consequently, the Auction algorithm appears to be much more flexible both regarding the number of tasks into which a job can be split and the number of clients that are operating in the environment. In Experiment B, we also compared performance with a random task assignment, which is computationally simple. As expected, the results indicate that our approach, which is capable of making informed decisions about where to distribute tasks, achieves better performance than a pure random approach. As we can conclude from the experiments presented above, in a MP2P WSN the physical network conditions have a major impact on the performance of job collaborations between peer nodes.

4.2 Implications for MP2P Systems

So far we have considered the base case of a network in which information about underlying radio conditions is available to the task distribution process. In a MP2P system, two things change: (i) task offload may still occur to nodes that are logical peers, but connected by a route of more than one physical link; (ii) information about the nature of this route may be unavailable to the application because of abstraction away from the physical network. Two potential solutions exist to this: (i) to adopt a cross-layered approach providing an API that permits the placement decision process in the application overlay to access low level information; (ii) to infer approximate information about the physical state of the network through tests that can be performed without layer violation. In both cases, we consider that throughput and latency measurements for the *whole route* will be used as a measure of congestion.

The greatest challenge for MP2P systems lies in utilising only information available at the application overlay to make determinations of levels of congestion. Since it is unreasonable to assume that the same networking technology is used throughout all paths, the application overlay may be only left with the periodic measurements of latency to estimate the available bandwidth between peer nodes. It is relatively straightforward to see that latency (or RTT, if clocks are not synchronised) can be measured directly simply by timestamping application overlay packets. Likewise, there are a range of techniques for estimating instantaneous bandwidth on routes, by using latency measurements [16, 12]. Such techniques are likely to provide adequate estimates for the bandwidth available in wired systems. However, they are not very adequate in MP2P WSNs.

For MP2P WSNs, it is important to distinguish between two effects that act on the latency along a path: the first effect is due to the length/number of hops of the path underlying a logical link between peers [11], and the second is the variability in the quality of that path as a result of congestion (as shown in Sect. 4.1). The fact that changes in latency due to congestion along a route occur at a frequency that exceeds those due to the changes in path length [18] can be exploited to separate the two effects in estimating the available bandwidth. For example, after obtaining a time series of latency measurements, a Hamming windowed FIR-H filter applied at, say, 1Hz, may separate the short timescale congestive effects on latency from the longer timescale effects of changing path length. This will establish more accurate upper and lower bounds for the available bandwidth and allow for a more effective estimate of likely bandwidth availability of the path that respects both effects. Such techniques of effective bandwidth estimation for MP2P WSNs are under our current investigation and will be proposed in a future paper.

5 Related Work

P2P WSN is a very new research field. Most work in this area deals with peer routing, searching and data query issues (e.g. [3], [14]). P2P overlay solutions were also presented to allow users to access various services across different physical sensor networks (e.g. [8], [14]). These works show the potential of better application support employing P2P overlay in WSNs. However, the key issue of P2P computing, i.e. load distribution in WSN, which is characterised by its heavily limited network layer, is yet to find a solution.

The problem of computation distribution has been extensively studied in the past in the grid computing area [7, 9, 15]. Regrettably, the assumptions behind most existing approaches render them largely unsuitable for use in networks of highly constrained devices (WSNs). Recent research has explored collaborative computation distribution in WSNs, but rarely in a P2P network model. Some of these approaches are inspired by the *client-server* based paradigm, others by the *mobile-agent* based one. Both can be combined with *cluster* based techniques. A range of approaches based on a cluster based computational model can be found in [19, 20, 10]. They all have in common the view that there is a hierarchical network architecture comprising of a high number of low-cost, less powerful sensors, and a small number of higher-cost, more powerful cluster heads. Such algorithms are particularly interesting if the structure of the deployed network is indeed hierarchical, and if the geographic distribution of cluster heads relative to sensor nodes is appropriate. However, cluster based approaches are inappropriate otherwise, and the existing approaches also fail to

take into account issues that relate to bandwidth utilisation, message collision and realistic radio modelling: communication is assumed to be collision-free. Furthermore, some of them [20] make two strong assumptions: firstly, when a certain event occurs, all sensor nodes can detect it and collect raw data; secondly, there are no events simultaneously occurring in the field. Moreover, tests are undertaken within oversimplified simulation environments that do not take into account real communication issues. Chiasserini et al. [4] propose the replication of an algorithm on every node and to split the data to compute among the peers, but they do not tackle the congestion problem. Abrams et al. in [2] study an optimisation algorithm for the assignment of tasks to microservers.

6 Conclusion

In this work, we presented a task load sharing scheme for MP2P WSNs. Experiments to evaluate the performance results were conducted on real Tmote Sky sensor testbeds, thus avoiding oversimplified radio models employed by the majority of simulation environments. In particular, we established that: (i) ignoring underlying network contention is highly unwise, in view of the magnitude of the effect on job execution time; (ii) simple load sharing algorithms can be adapted to take into account both computational capabilities and network conditions, with the result of achieving execution times that approximate those of an uncongested system; and (iii) there are plausible approaches for applying this technique for use in MP2P systems.

7 Acknowledgments

We would like to thank Björn Grönvall and all the Contiki OS team for the technical support, Wolfgang Emmerich and Ettore Ferranti. This work is part of the Divergent Grid project funded by EPSRC under grant number EP/C534891.

References

[1] Heterogeneous Experimental Network (HEN) - University College London. http://www.cs.ucl.ac.uk/research/hen/.

[2] Z. Abrams, H.-L. Chen, L. Guibas, J. Liu, and F. Zhao. Kinetically Stable Task Assignment for Networks of Microservers. In *Proceedings of the* 5^{th} *ACM/IEEE IPSN*, pages 93–101, April 2006.

[3] M. Ali and K. Langendoen. A Case for Peer-to-Peer Network Overlays in Sensor Networks. In *Proceedings of WWSNA with* 6^{th} *IPSN*, April 2007.

[4] C. F. Chiasserini and R. R. Rao. On the Concept of Distributed Digital Signal Processing in Wireless Sensor Networks. In *Proceedings of the IEEE/Boeing MILCOM Conference*, pages 260–264, October 2002.

[5] P.-J. Chuang and C.-W. Cheng. On File and Task Placements and Dynamic Load Balancing in Distributed Systems. *Tamkang Journal of Science and Software Engineering*, 5(4):241–252, 2002.

[6] A. Dunkels, O. Schmidt, T. Voigt, and M. Ali. Protothreads: Simplifying Event-Driven Programming of Memory-Constrained Embedded Systems. In *Proceedings of the* 4^{th} *ACM SenSys*, pages 29–42, November 2006.

[7] D. L. Eager, E. D. Lazowska, and J. Zahorjan. Adaptive Load Sharing in Homogeneous Distributed Systems. *IEEE Transaction on Software Engineering*, 12(5):662 – 675, May 1986.

[8] H.-W. Fang, L.-C. Ko, and H.-Y. Fang;. A Design of Service-based P2P Mobile Sensor Networks. In *Proceedings of IEEE SUTC*, pages 152–157, June 2006.

[9] D. Ferguson, Y. Yemini, and C. Nikolaou. Microeconomic Algorithms for Load Balancing in Distributed systems. In *Proceedings of the* 8^{th} *IEEE ICDCS*, pages 491–499, June 1988.

[10] A. Fuggetta, G. P. Picco, and G. Vigna. Understanding Code Mobility. In *Proceedings of the IEEE Transaction on Software Engineering*, volume 24, pages 342–361, May 1998.

[11] P. Gupta and P. Kumar. The Capacity of Wireless Networks. *IEEE Transaction on Information Theory*, 46(2):388–404, March 2000.

[12] N. Hu and P. Steenkiste. Exploiting Internet Route Sharing for Large Scale Available Bandwidth Estimation. In *Proceedings of the IMC*, pages 187–192, October 2005.

[13] D. Kotz, C. Newport, R. S. Gray, J. Liu, Y. Yuan, and C. Elliott. Experimental Evaluation of Wireless Simulation Assumptions. Technical Report TR2004-507, June 2004.

[14] S. Krco, D. Cleary, and D. Parker. P2P Mobile Sensor Networks. In *Proceedings of the* 38^{th} *HICSS*, pages 324c–324c, January 2005.

[15] S.-M. Lau, Q. Lu, and K.-S. Leung. Adaptive Load Distribution Algorithms for Heterogeneous Distributed Systems with Multiple Task Classes. *Journal of Parallel and Distributed Computing, ELSEVIER*, 66(2):163–180, February 2006.

[16] X. Liu, K. Ravindran, and D. Loguinov. Multi-Hop Probing Asymptotics in Available Bandwidth Estimation: Stochastic Analysis. In *Proceedings of the IMC*, pages 173–186, October 2005.

[17] L. Mottola and G. P. Picco. Programming Wireless Sensor Networks with Logical Neighborhoods. In *Proceedings of the* 1^{st} *ACM InterSense*, May 2006.

[18] M. Musolesi, S. Hailes, and C. Mascolo. Adaptive Routing for Intermittently Connected Mobile Ad Hoc Networks. In *Proceedings of the* 6^{th} *IEEE International WoWMoM*, pages 183–189, June 2005.

[19] M. Singh and V. K. Prasanna. A Hierarchical Model for Distributed Collaborative Computation in Wireless Sensor Networks. In *Proceedings of the* 17^{th} *IEEE IPDPS*, pages 166–176, April 2003.

[20] Y. Xu and H. Qi. Distributed Computing Paradigms for Collaborative Signal and Information Processing in Sensor Networks. *Journal of Parallel and Distributed Computing, ACM*, 64(8):945–959, August 2004.