

Community Sensor Grids: Virtualization for Sharing across Domains

Amiya Bhattacharya
Department of Computer Science
New Mexico State University
Las Cruces, NM 88003
amiya@nmsu.edu

Meddage S. Fernando
Department of Computer Science
New Mexico State University
Las Cruces, NM 88003
saliya@nmsu.edu

Partha Dasgupta
School of Computing and Informatics
Arizona State University
Tempe, AZ 85287
partha@asu.edu

ABSTRACT

Wireless sensor networks have been traditionally designed to be privately owned and used. Hence the two hallmark features of sensor networks, namely customized network applications and the collaborative in-network processing, are not achievable beyond the boundary of the users' administrative domains (except for limited scope data sharing through Internet gateways). This position paper advocates a contrarian notion of sensor grid that preserves the operational continuity of the participating sensor networks and yet allows controlled sharing of sensor node hardware capabilities over multiple administrative domains. This is achieved by stitching together virtualized nodes donated by participating sensor networks into a transient virtual sensor network underlay, which is called a Community Sensor Grid. Possible approaches for node virtualization support are discussed for some existing sensor network OS platforms. The formation, operation and dissolution of the transient underlay are to be supported by a P2P overlay formed by the Internet gateways of participating sensor networks.

Categories and Subject Descriptors

C.2.1 [Computer-Communication Networks]: Network Architecture and Design—packet-switching networks, store and forward networks, wireless communication.

C.3 [Special-purpose and Application-based System]: Real-time and embedded systems.

General Terms

Design, Performance, Security, Standardization, Management.

Keywords

Wireless PAN, Mote virtualization, Concurrency, Internet integration, Overlay and underlay networks.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MobiVirt'08, June 17, 2008, Breckenridge, CO.
Copyright 2008 ACM 978-1-60558-328-0/08/06...\$5.00.

1. INTRODUCTION

Wireless Sensor Networks (WSNs or sensor networks), a class of self-organizing wireless ad-hoc networks made of sensor nodes (also known as motes), are considered to be “a new family of networks” primarily due to two reasons. First, an individual sensor node is resource-constrained in terms of processing power, storage, bandwidth and battery life—but the network, being composed of a large number of such nodes, offers substantial processing capability and battery life as a whole [7]. Second, nodes perform both in-situ and in-network processing while propagating queries and responses to make the network more energy-efficient than one using a centralized scatter-gather approach [25][26].

A sensor network has a commonly accepted structure of being a cloud of sensor nodes hanging off of a gateway at the edge of the Internet. In the context of the overall Internet architecture, it exists as a collection of one or more “privately owned” subnets, under a single administrative domain [20]. While being under a single administrative authority is particularly suited for application-specific programming of processing and forwarding functionalities at every node, one needs complete control of the sensor network. This, in turn, translates to ownership.

A downside of private ownership is that it does not lend itself easily to a variety of usage scenarios, especially community-based sharing. If a community accessible sensor grid across multiple administrative domains is desired, the current state-of-the-art is to interconnect several isolated sensor network clouds with Internet gateways—using published services as building blocks for data-only accesses.

In not too distant future, sensor nodes are likely to be deployed in various capabilities by private citizens, organizations, governments and social groups alike. While private use such as securing property and monitoring assets may be the primary reasons behind these deployments, their ubiquitous presence and span offer a great potential for community use (beyond the obvious private use situations). As seen in Figure 1, an appealing possibility is where a user with proper credentials can join a peer-to-peer (P2P) network to discover available sensor nodes with advertised capabilities in an intended area of coverage—thereby forming an overlay network using a subset of these nodes to inject a new sensing task without disturbing the ongoing primary tasks at any node.

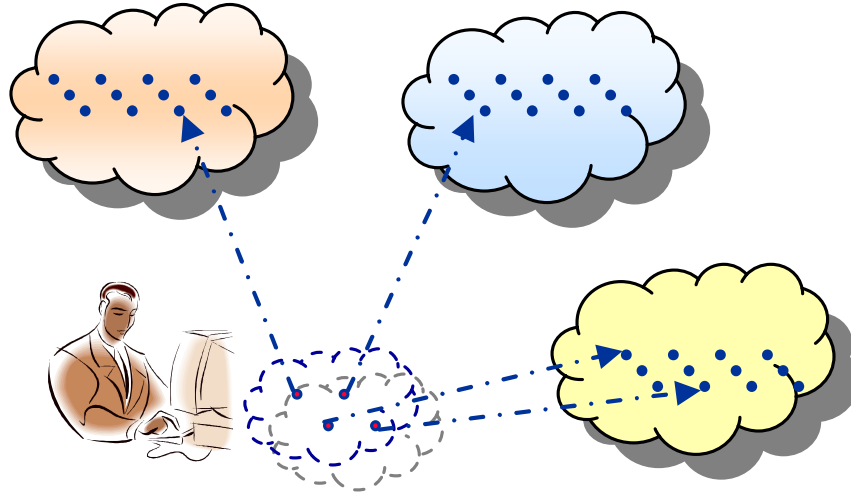


Figure 1: Virtual sensing/actuating service provides naming, discovery, custom applications and in-network processing.

In this position paper, we introduce the concept of a *community sensor grid*, which is a virtual sensor network built using parts of several physical sensor networks. The community sensor grid is a *transient* sensor network made of nodes with inherent heterogeneity in ownership, hardware-software platforms, and exposed capabilities. A sensor grid is composed of nodes contributed by more than one private (physical) sensor networks, all having Internet gateways which participate in a P2P network to offer processing view of virtual sensor nodes based on a common sharing semantics. We address the issues of deployment, scalability, security and topology management.

2. MODELS OF SHARING

The majority of the efforts towards Internet integration of sensor networks to date rely on a data-only sharing model. A sensor network is often conceived as a low-power WPAN (Wireless Personal Area Network) topologically, while as a database or file system from an application-level point of view [6][11]. An Internet client program usually interacts with this application-level processing view of the sensor network through a proxy server. The proxy service usually runs at an IP-enabled 32-bit master node or micro-server which controls the nodes that belong to the WPAN. The sensing capabilities of the nodes are often pre-determined and advertised by the micro-servers handling queries. Enabling IP protocol stack on nodes is getting increasingly popular due to usability of existing client programs that are based on TCP, UDP or SNMP [1][8]. Middleware support in some systems allows the Internet client programs to control operational parameters such as sampling intervals and reporting filters at the nodes [2]. Even real-time streaming of sensor data for live feed to the computational grid [19] or publishing to the Web [23] concentrates only on the relevant APIs for interacting with the proxies, supporting only the data-only sharing model.

In effect, private sensor networks offer only a limited form of external programmability support (such as altering some parameters). Arguments in favor of this design may be that publishing sensor readings is a service usually sufficient for traditional use of a

sensor network, while not requiring power-consuming over-the-air reprogramming of nodes.

A more flexible level of programmability is however offered externally under a very different usage scenario where the system designer is perpetually unaware about the users' need. Community-shared sensor network research testbeds such as MoteLab [28] and Kansei [10] allow remote users to program a sensor network using a Web-based interface, so as to run an experiment during a reserved slot scheduled ahead of time. While the experiments run on the wireless data plane, these sensor network infrastructures typically offer an Ethernet-based wired control plane for programming and data collection. Building a set of pre-determined services is never a viable alternative for testbed design, as it sacrifices flexibility for experimentation.

In summary, we can conclude that, (i) the majority of the external access mechanisms support a data-only access model in sensor networks, and (ii) total access model, whenever supported, is exclusively enforced by reservation.

3. SENSOR NETWORK OF VIRTUAL NODES

Let us first explore what a new model of sharing, namely non-exclusive total access, has to offer.

3.1 Motivation

We envision that home security kits built around WPAN-based nodes will be widely available for use in the near future. Moreover, convergence to a single MAC/PHY standard (e.g., IEEE 802.15.4) also seems inevitable for inter-operability of products from different manufacturers. Such private deployments, however, will vary widely in terms of network size. A small deployment of a few nodes will typically form a single hop star topology around the Internet gateway, while a medium size network will self-organize in the form of the familiar multihop mesh to maximize coverage. Larger deployments may call for a tiered organization around more than one IP-enabled master nodes connecting to the Internet gateway.

Let us now explore how one can temporarily “borrow” several nodes from several separate domains, virtual sensornet that extends the area of coverage beyond some physical boundary. A non-exclusive total access model of sharing would be quite appropriate. Also, as the area under coverage may span over both private and public properties, privacy rights of participating individuals must be reflected by due assignment of sensing and monitoring capabilities.

Our motivation is to explore the possibilities of forming scalable community-based transient sensornets based on a non-exclusive total access model, so that ordinary citizens are empowered to program networked sensing applications expanding the reach beyond their ownership. In what follows, we attempt to present a scalable architecture under the following design assumptions:

1. A community of users (private, corporate and/or government) is willing to deploy sensor node hardware to cover areas under their jurisdiction. The covered areas may be isolated to start with.
2. There exists a level of willingness from all members of the community to share the sensing and processing capabilities of their sensornet infrastructure as needed, without interrupting the primary sensing task for the respective deployments.
3. A typical user belonging to the community often borrows sensor node capabilities from parts of the community sensornet infrastructure to form a transient virtual sensornet over a desired area of coverage. The transient network can be dissolved after the task completes and results are gathered.
4. The rules of sharing are dictated by the owner of the resource and at no time the sensing task of a local or remote user violates the authority, privacy and security of the provider.
5. Optionally, a supporting business model accounts for balancing the cost of sharing resources (such as hardware and battery costs) between private and community use.

3.2 A Case for Mote-level Virtualization

As stated above, we are driven by the envisioned availability of WPAN deployments to build a scalable architecture for non-exclusive total access to sensornets from outside their administrative domains. The need for virtualization at the mote level arises from the non-exclusive total access requirement, as the primary sensing task of the owner must co-exist with the newly injected sensing task for the remote user. To contrast with the terminologies in OS-level virtualization in the multitasking workstation/server environments, we opt for calling them the host task and the guest task respectively.

In the simplest scenario, there are only one host task and at most one guest task executing in a mote (in the general case there may be more guests, up to a resource limited maximum). In the macroscopic view of time, a mote is thus capable of participating in more than one sensornets simultaneously. The host task participates in forming the host network, while the guest task forms the guest network. Naturally, the mote must maintain two different network identities for transmitting and receiving packets over the radio. Among the two overlapped wireless networks, the host network strictly observes the boundary enforced by ownership. In other words, a mote with the host network identity cannot communicate with another mote outside the host network. A virtualized mote with the guest network identity is allowed to

communicate with another virtualized mote across the ownership boundaries as long as both virtualized motes are participating in the same guest network. The host networks ensure preservation of the primary activities within all the deployed motes, whereas the guest network is formed to share community resources. Figure 2 shows how several virtualized motes from multiple host sensornets are stitched together to blur domain boundaries while forming a guest sensornet.

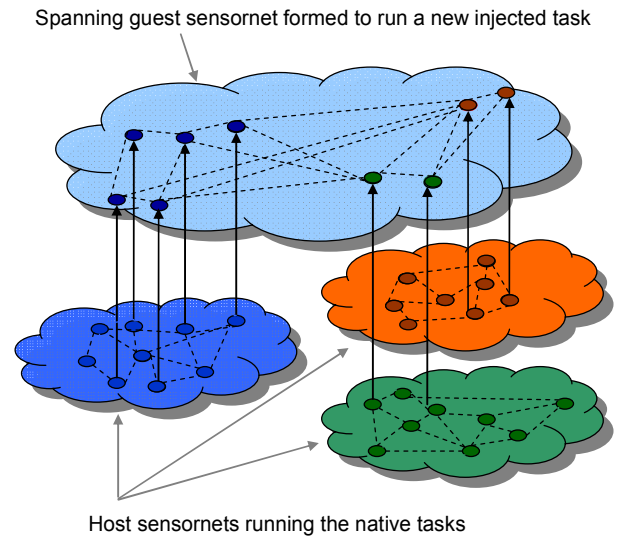


Figure 2: Formation of a guest sensornet with expanded boundary using mote-level virtualization

To the best of our knowledge, the design goals stated here have not yet been addressed by the networked sensing research community. However, it is worth noting that the salient theme is reminiscent of the resource sharing in the PlanetLab (<http://www.planet-lab.org>) infrastructure for distributed computing and the GENI (<http://www.geni.net>) infrastructure under development for future Internet design. While we understand that it is a significant challenge to implement virtualization on resource constrained platforms such as the motes, we postpone these discussions until the next section. Instead, we present an example to motivate the benefits and a business model associated with such type of community-based sharing.

Let us consider a scenario in urban sensing, where a city is interested in improving its constant vigilance on brush fire to actuate an early response system. City authority has property rights only to the streets and public places for sensornet deployment, but no access to private properties for sensor deployment. However, citizens can be enticed to deploy mote-based sensornets within their property at their own cost, and have tax rebates for allowing the city guest access. The problem is not solved efficiently by periodic or queried data-only access, as the application requires adaptive sampling and extensive use of single hop exchange of temperature readings among neighbors to detect the spreading contour. To utilize the sensor nodes’ computing power for adaptive sampling, city engineers need total access of the sensor motes. Data exchange with neighbors is allowed in the

city's guest network, which spans beyond the boundaries of the private properties. In absence of such a guest network, neighboring motes on two sides of a property boundary line would not have participated in data exchange as neighbors, and the application had to use the data-only model of sensing, executing the contour detection algorithm at a remote centralized location.

4. SENSOR NODE VIRTUALIZATION

Due to the constant pressure of keeping the systems footprint low, embedded operating systems and middleware platforms for sensornets have wide variations in providing the building blocks needed for mote level virtualization. To be successful, an architectural solution such as the one we present here should ideally be supported on most popular platforms. Unfortunately, since most platforms are research prototypes under active development, it is a challenge to do a thorough comparative capability evaluation of even the most dominant platforms. We present here some findings from our continuing survey.

4.1 Concurrency Support

Programming sensornets has traditionally been under an event-driven framework, such as in TinyOS, the de facto standard platform [17]. TinyOS supports non-preemptive scheduling of tasks, which can only be preempted by events. The underlying assumption is that, due to low duty cycle of tasks (an implicit assumption for sensornets anyway), the lack of preemption capabilities would not cause perceivable performance penalty on concurrency seen at a macroscopic level. Other operating systems, such as Contiki and SOS, also conform to this philosophy [9][12].

A traditional thread-based approach to concurrency is supported by the kernel of MANTIS OS, which offers preemptive scheduling [3]. However, there exist efforts to implement thread abstractions on operating systems primarily supporting event-driven tasking—with the assumption of no visible performance penalty for low duty cycle tasks [21]. Running distinct host and guest threads is particularly simple on platforms with the thread support, as demonstrated by the TinyMOS implementation, which can run a typical TinyOS program as a MANTIS thread [27].

4.2 Supporting Host and Guest Networks

A mote, having the host and the guest threads running concurrently, participates in two (or more) distinct sensornets—namely the host and the guest nets. Even though the guest threads may be considered to create one or more virtualized mote entity, the guest sensornets are all real. To be more specific, there are new WPANs formed by the virtualized motes carrying frames with headers that differ in either the channel ID or the network (or group) ID from the frames of the original host WPANs.

Thus, there is a need for explicitly changing channel, network-id and host-id at every transition of control between the host and the guest threads. While changing channel, network, or node id's for a mote is not common between wired reprogramming, the operating systems offer explicit support for this, which has been used for purposes such as throughput optimization [15].

Switching between networks is possibly going to be the prime contributor to the performance loss due to mote virtualization. First, changing channel has a longer delay (of the order of

milliseconds) in comparison to changing WPAN id or node id, as the later are implemented in software as a filter. However, channel switching cannot be avoided because the guest sensornet spans over multiple host sensornets, and the host sensornets can natively be on different channels to start with. Channel assignment to the guest sensornet can still be done so as to let the least number of motes to suffer the time lag due to channel switching. The second contributor to performance loss is the deafness imposed when the radio is tuned to the other channel. None of the existing mote hardware platforms supports a radio implementing any multi-channel MAC protocol. At this time, the problem needs to be handled entirely by co-ordination of motes based on time-synchronization. However, other wireless standards such as IEEE 802.11 have multi-channel MAC protocol implementations, which may be available in the mote platform in the future [15].

4.3 Runtime Injection and Protection

TinyOS offers minimal support for injecting a new task to be run concurrently without stopping the current one. Achieving this capability under TinyOS requires early decision on the components that will be used by the injected task. These components can then be statically compiled as part of the host task. Using the middleware support of a virtual machine such as Maté [16], one can then propagate the guest task as byte code that later invokes execution of the required components [29]. While supporting a similar event-based programming framework, SOS [12] attempts to support runtime binding of dynamic components based on software memory protection [14]. In addition, SOS offers a virtual machine called DVM that can accept dynamic components as well, without prior decision made at the compilation of the native host task [2]. Byte codes are typically propagated as epidemic [18], whereas frequent full over-the-air reprogramming is more power hungry [13].

While running distinct host and guest threads under a platform like MANTIS is appealing for concurrency, byte code implementation of the guests has some advantages in enforcing access control by sandboxing. Capabilities such as switching channel, network, or host id should be owned by the host system only. Explicit steps are to be taken to prevent guest threads from listening in promiscuous mode or scanning channels, so as to protect the privacy of host's native sensing tasks.

5. INTERNET INTEGRATION

It is well understood that the narrow waist of sensornet protocols is a translucent abstraction of MAC layers. TinyOS 2 implements it explicitly as a layer called SP [24]. We believe that this the most ideal layer to implement mote virtualization. Since this lies below the IPv6 address assignment as per the 6LoWPAN standard [1][22], we refer to the guest sensornet as an underlay. The underlay needs to incorporate security solutions using shared keys between gateway and the motes similar to the traditional sensor networks, thus providing both privacy and authentication.

Every participant in the community sensor grid must have a sensornet with an Internet gateway. The gateway forms the point of presence of the internal sensornets on the Internet, and is responsible for exposing the sensornet capabilities. The gateways must register themselves to form a P2P network where resources and associated usage policies can be discovered by other peers. Users access this P2P network via an "account manager," which

has a public-key based certificate. When a user wants to create a private network, the account manager negotiates the level of service with each gateway that controls a participating sensornet. The negotiation is based on the account manager's certificate and the gateway's security policy. A user can subsequently form another overlay network consisting of all the gateways to the sensornet he/she wants to use. After such negotiations are completed, the account manager and the gateways exchange a group session key and then the group session key is made available to the user application, which can then proceed to pass on the program to inject and collect sensor readings. The overlay network must be set up before the community sensornet underlay (guest sensornet) is formed, and must remain until the underlay is dissolved. In case a sensornet gateway leaves the overlay with or without notification, the community sensornet underlay needs to be reformed.

6. RELATED WORK

Our approach towards forming community sensor grids falls under the general area of participatory sensing, which is emerging to be an active area of research in both academia and industry. Naturally, the design we propose here shares some of the design goals that are common to these research projects, while addressing unique problems that are complementary in nature.

The goal of the urban participatory sensing project at the Center for Embedded Networked Sensing (CENS) is to engage sensors built into portable devices such as mobile phones and PDAs in sharing their capabilities [4]. A tiered architecture and related protocols for people-centric urban sensing is under development as part of the MetroSense project at Dartmouth, which involves mobile devices in opportunistic interactions with wireless Internet gateways and available static sensornet infrastructure [5].

The community sensor grid architecture is aimed at enabling private sensornet infrastructures for participatory sensing. While using or supporting mobile devices is outside its scope at this time, the proposed P2P based control is amenable to extensions that will support mobile control stations or sinks.

One important difference between the community sensor grid and most other participatory sensing projects is the sharing model supported by the underlying architecture. Building a set of predefined services into the participating entities usually suffices in most cases, as long as the target applications can be developed based on data-only sharing. In contrast, we attempt to dynamically load and run foreign applications in an existing and operational sensornet. This makes concurrency, reprogramming and authentication to be the central theme.

7. CONCLUSION AND ONGOING WORK

This paper presents a concept of a community sensor grid that allows the sharing of the sensing and processing facilities of motes across many domains, each domain being a traditional sensornet. The sharing allows guests to participate in sensing activities (access controls apply) using sensornets deployed by several different organizations or private owners. The result is a virtual sensornet, that is transient and with inherent heterogeneity in ownership, hardware/software, and exposed capabilities. The Internet gateways of the underlying physical sensornets form a

P2P overlay network. We address some issues of deployment, scalability, security and topology management.

Our ongoing work is to realize the above architecture to enable deployment of virtual sensornets of community sensor grids. The deployment issues include mote virtualization (with guest and host applications), dynamic application invocation and termination, network overlay and underlay formation, Internet gateway management and access control. We are in the process of designing a prototype implementation using motes and gateways.

8. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grants no. CNS-0551734 and CNS-0617671, as well as a Graduate Research Enhancement Grant from the Office of Vice-President of Research, Graduate Studies, and International Programs at the New Mexico State University.

9. REFERENCES

- [1] Arch Rock Corporation, 2007. A sensor network architecture for the IP enterprise. In proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN 2007, Cambridge, Massachusetts, April 2007), 575. DOI=[10.1145/1236360.1236447](https://doi.org/10.1145/1236360.1236447).
- [2] R. Balani et al., 2006. Multi-level software reconfiguration for sensor networks. In proceedings of the 6th ACM/IEEE International Conference on Embedded Software (EMSOFT'06, Seoul, Korea, October 2006), 112-121. DOI=[10.1145/1176887.1176904](https://doi.org/10.1145/1176887.1176904).
- [3] S. Bhatti et al., 2005. MANTIS OS: An embedded multi-threaded operating system for wireless micro-sensor platforms. *Mobile Networks and Applications*, 10, 4 (August 2005), 563-579, Springer Science. DOI=[10.1145/1160162.1160178](https://doi.org/10.1145/1160162.1160178).
- [4] J. Burke et al., 2006. Participatory sensing. In proceedings of the 1st Workshop on World-Sensor-Web (WSW'06, Boulder, Colorado, October 2006).
- [5] A. Campbell et al., 2006. People-centric urban sensing. In proceedings of the 2nd Annual International Workshop on Wireless Internet (WICON 2006, Boston, Massachusetts, August 2006). DOI=[10.1145/1234161.1234179](https://doi.org/10.1145/1234161.1234179).
- [6] Q. Cao et al., 2008. The LiteOS Operating System: Towards Unix-like abstractions for wireless sensor networks. In proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN 2008, St. Louis, Missouri, April 2008), 233-244. DOI=[10.1109/IPSN.2008.54](https://doi.org/10.1109/IPSN.2008.54).
- [7] D. Culler, D. Estrin and M. Srivastava, 2004. Overview of sensor networks. *IEEE Computer*, 37, 8 (August 2004), 41-49. DOI=[10.1109/MC.2004.93](https://doi.org/10.1109/MC.2004.93).
- [8] A. Dunkels et al., 2003. Full TCP/IP for 8-bit architectures. In proceedings of the 1st International Conference on Mobile Systems Applications and Services (MobiSys '03, San Francisco, California, May 2003), 85-98. DOI=[10.1145/1066116.1066118](https://doi.org/10.1145/1066116.1066118).
- [9] A. Dunkels et al., 2006. Run-time dynamic linking for reprogramming wireless sensor networks. In proceedings of

- the 4th International Conference on Embedded Networked Sensor Systems (SenSys'06, Boulder, Colorado, November 2006), 15-28. DOI=[10.1145/1182807.1182810](https://doi.org/10.1145/1182807.1182810).
- [10] E. Ertin et al., 2006. Kansei: A testbed for sensing at scale. In proceedings of the 5th International Conference on Information Processing in Sensor Networks (IPSN 2006, Nashville, Tennessee, April 2006), 399-406. DOI=[10.1145/1127777.1127838](https://doi.org/10.1145/1127777.1127838).
- [11] J. Gehrke and S. Madden, 2004. Query processing in sensor networks. *IEEE Pervasive Computing*, 3, 1 (January-March 2004), 46-55. DOI=[10.1109/MPRV.2004.1269131](https://doi.org/10.1109/MPRV.2004.1269131).
- [12] C.-C. Han et al., 2005. A dynamic operating systems for sensor nodes. In proceedings of the 3rd International Conference on Mobile Systems Applications and Services (MobiSys'05, Seattle, Washington, June 2005), 163-176. DOI=[10.1145/1067170.1067188](https://doi.org/10.1145/1067170.1067188).
- [13] J. Hui and D. Culler, 2004. The dynamic behavior of a data dissemination protocol for network programming at scale. In proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04, Baltimore, Maryland, November 2004), 81-94. DOI=[10.1145/1031495.1031506](https://doi.org/10.1145/1031495.1031506).
- [14] R. Kumar, E. Kohler and M. Srivastava, 2007. Harbor: Software-based memory protection for sensor nodes. In proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN 2007, Cambridge, Massachusetts, April 2007), 340-348. DOI=[10.1145/1236360.1236404](https://doi.org/10.1145/1236360.1236404).
- [15] H. K. Le, D. Henriksson and T. Abdelzaher, 2007. A control theory approach to throughput optimization in multi-channel collection sensor networks. In proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN 2007, Cambridge, Massachusetts, April 2007), 31-40. DOI=[10.1145/1236360.1236365](https://doi.org/10.1145/1236360.1236365).
- [16] P. Levis and D. Culler, 2002. Maté: A tiny virtual machine for sensor networks. In proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS X, San Jose, California, October 2002), 85-95. DOI=[10.1145/605397.605407](https://doi.org/10.1145/605397.605407).
- [17] P. Levis et al., 2004. The emergence of networking abstractions and techniques in TinyOS. In proceedings of the 1st Symposium on Networked Systems Design and Implementation (NSDI'04, San Francisco, California, March 2004), 1-14.
- [18] P. Levis et al., 2004. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor network. In proceedings of the 1st Symposium on Networked Systems design and Implementation (NSDI'04, San Francisco, California, March 2004), 15-28.
- [19] H. B. Lim et al., 2007. The national weather sensing grid. In proceedings of the 6th International Conference on Embedded Networked Sensor Systems (SenSys'07, Cambridge, Massachusetts, April 2007), 369-370. DOI=[10.1145/1322263.1322299](https://doi.org/10.1145/1322263.1322299).
- [20] K. Mayer and W. Fritsche, 2006. IP-enabled wireless sensor networks and their integration into the Internet. In proceedings of the 1st International Conference on Integrated Internet Ad hoc and Sensor Networks (InterSense'06, Nice, France, May 2006). DOI=[10.1145/1142680.1142687](https://doi.org/10.1145/1142680.1142687).
- [21] W. P. McCartney and M. Sridhar, 2006. Abstractions for safe concurrent programming in networked embedded systems. In proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys'06, Boulder, Colorado, November 2006), 167-180. DOI=[10.1145/1182807.1182825](https://doi.org/10.1145/1182807.1182825).
- [22] G. Montenegro et al., 2007. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. The Internet Engineering Task Force (IETF) RFC 4944, September 2007.
- [23] S. Nath et al., 2006. SensorMap: A web site for sensors world-wide. In proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys'06, Boulder, Colorado, November 2006), 373-374. DOI=[10.1145/1182807.1182861](https://doi.org/10.1145/1182807.1182861).
- [24] J. Polastre et al., 2005. A unifying link abstraction for wireless sensor networks. In proceedings of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys'05, San Diego, California, November 2005), 76-89. DOI=[10.1145/1098918.1098928](https://doi.org/10.1145/1098918.1098928).
- [25] G. J. Pottie and W. J. Kaiser, 2000. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51-58, (May 2000). DOI=[10.1145/332833.332838](https://doi.org/10.1145/332833.332838).
- [26] K. Sohrabi et al., 2000. Protocols for self-organization of a wireless sensor networks. *IEEE Personal Communications*, 7, 5(October 2000), 16-27. DOI=[10.1109/98.878532](https://doi.org/10.1109/98.878532).
- [27] E. Trumpler and R. Han, 2006. A systematic framework for evolving TinyOS. In proceedings of the 3rd Workshop on Embedded Networked Sensors (EmNets 2006, Cambridge, Massachusetts, May 2006).
- [28] G. Werner-Allen, P. Swieskowski and M. Welsh, 2005. MoteLab: A wireless sensor network testbed. In proceedings of the 4th International Conference on Information Processing in Sensor Networks (IPSN'05, Los Angeles, California, April 2005), 483-488. DOI=[10.1109/IPSN.2005.1440979](https://doi.org/10.1109/IPSN.2005.1440979).
- [29] Y. Yu et al., 2006. Supporting concurrent applications in wireless sensor networks. In proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys'06, Boulder, Colorado, November 2006), 139-152. DOI=[10.1145/1182807.1182822](https://doi.org/10.1145/1182807.1182822).