Towards a Sensor Network Architecture: Lowering the Waistline

David Culler*, Prabal Dutta*, Cheng Tien Ee*, Rodrigo Fonseca*,
Jonathan Hui*, Philip Levis*, Joseph Polastre*, Scott Shenker* †,
Ion Stoica*, Gillman Tolle*, and Jerry Zhao†

1. INTRODUCTION

Wireless sensor networks have the potential for tremendous societal benefit by enabling new science, better engineering, improved productivity, and enhanced security. Research in this area has progressed dramatically in the past decade. The hardware, particularly radio technology, is improving rapidly, leading to cheaper, faster, smaller, and longer-lasting nodes. Many systems challenges, such robust multihop routing, effective power management, precise time synchronization, and efficient in-network query processing, have been tackled and several complete applications, in which all these components have been integrated into a coherent system, have been deployed and demonstrated, including some at relatively large scale [?, ?].

But the situation in sensornets, while promising, also has problems. The literature presents an alphabet soup of protocols and subsystems that make widely differing assumptions about the rest of the system and how its parts should interact. The extent to which these components can be combined to build usable systems is quite limited. In order to produce running systems, various research groups have produced "vertically integrated" designs in which their own set of components are specifically designed to work together, but are unable to interoperate with components from other groups. This greatly reduces the synergy between research efforts, and has impeded progress in the field.

It is the central tenet of this paper that the primary factor currently limiting progress in sensornets is not any specific technical challenge (though many remain, and deserve much further study) but is instead *the lack of an overall sensor network architecture*. Such an architecture would identify the essential components and their conceptual relationships so that it would become possible to compose components in a manner that transcends particular generations of technology, allows innovation, and promotes interoperability.

*CS Division, UC Berkeley, Berkeley, CA 94720.
†ICSI, 1947 Center St., Berkeley, CA 94704

2. THE NATURE OF AN ARCHITECTURE

One goal of an architecture is to allow components developed for one system to be used in many other systems, especially those from other developers. However, the need to build composable components is hardly unique to sensornets; it is a standard challenge in building large software systems. The challenge in sensor networks is that applications are meaningfully distributed over a large number of relatively constrained nodes that are embedded in physical space. The application dictates the sensor modalities and sample rates, the real-time processing and storage of this data, and the information exchange protocols among collections of nodes. Thus, the traditional divisions of application, operating system, and network have been relaxed.

This relaxation has led to re-examination [?] of the issues of scheduling, power-control, and information flow that cut across the traditional boundaries. Cutting across boundaries, however, easily leads to monolithic solutions or to subsystem components with arbitrary interface assumptions. For future work to be able to build on prior efforts, We need to re-establish a meaningful separation of concerns.

An architecture is a set of principles that guide where functionality should be implemented along with a set of interfaces, functional components, protocols, and physical hardware that (roughly) follows those guidelines. For instance, the Internet architecture demonstrated how a properly chosen set of guiding principles can shape the evolution of a complex system over vast changes in technology, scale, and usage [?]. The philosophy of designing for heterogeneity, change and uncertainty was a radical shift from classical systems design, which more traditionally seeks a near optimal assembly of near optimal parts. Faced with integrating several existing networks with widely varying characteristics, the end-to-end principle and focus on interoperability led to a design that has successfully coped with unprecedented scale, incorporated a vast array of new underlying technologies, and supported a dizzying variety of applications. However, it was not free of costs; the use of rigid layering sacrificed efficiency in various regards in return for increased interoperability.

The power of the Internet is revealed not so much in the elegance or efficiency of its individual components, but in the overall ability to encompass tremendous growth in scale and in diversity as usage and technology rapidly evolved. This is our goal for developing an architecture for sensornets. We must be extremely mindful of any loss of efficiency for particular tasks as we seek to greatly enhance the interoperability between components and ability to advance.

3. THE NARROW WAIST

A complete sensornet architecture will need to address a family of specific issues, such as discovery, topology management, naming, routing and so on, but the over-riding question is whether there is "narrow waist" — a unifying abstraction that permits a wide variety of uses above and a range of implementations below. At what level should it occur and what should it express? By requiring all network technologies to support IP, and all applications to run on top of IP, the Internet could accommodate, even encourage, a vast degree of heterogeneity and diversity in both applications and underlying technologies.[1] We have an analogous goal for sensornets; in both the application and device arenas we are in the midst of extremely rapid developments. Sensornets will only flourish if we can identify a narrow waist in the architecture that will allow device and protocol developments to proceed apace, while permitting significant optimization.

We claim that sensor networks can also have a narrow waist – the Sensor-net Protocol (SP) – and that it should be a best-effort single-hop broadcast with a rich enough interface to allow multiple network layer components above to optimize for a range of potential link layers below in a hardware-independent fashion.[2] It is not only the resource limitations of sensornets that cause the SP to be closer in nature to the data link layer than the network layer at which IP resides. Applications differ dramatically in their communication patterns and are intimately tied to their associated network protocols. They generally do not require and often do not benefit from a common, universally routable addressing scheme. Instead, a single, simple interface needs be able to efficiently implement a range of routing protocols independently of the underlying link layer, and to facilitate in-network processing and collective communication, as well as point-to-point transport. Moving the point of universal abstraction downward presents new issues that we do not typically concern ourselves about in the Internet architecture. It also requires a careful design of the layers above SP to provide a reasonably general platform on which to build various sensor network applications efficiently. If SP is to be a unifying abstraction with a common format and semantics across many physical layers, how functionality divides across the packet boundary is a key question.

We believe that the interface needs to be more expressive than current data link interfaces for higher level control, yet have decomposable functionality below for greater flexibility. To support the network protocols found in the sensornet literature, the mechanisms which a sender

[1]It is straightforward to incorporate sensornets as edge networks of the Internet with gateway nodes providing a bridge. IPv6 addressing makes this considerably easier. In the vast majority of cases, the gateway will also serve as a proxy, so TCP connections would rarely terminate at the actual sensor node. In the proxy case, it is also natural for collections of nodes to appear as a virtual Internet host. The most challenging question is the architecture within the sensornet. This is more than just another subnet, because distributed applications are spread over the many nodes in a manner dependent on its physical embedment.

[2]While IEEE 802.2 provides a link interface to a variety of links, higher layers must know what physical layer is below and access the specific physical layer through a standardized set of calls.

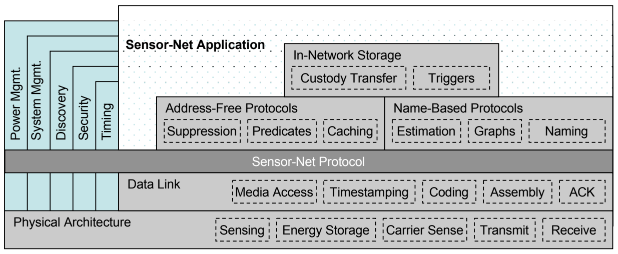

Figure 1: Sensor Network Functional Layer Decomposition

should be able to control including generation of link level acknowledgments, performance of initial and collision-avoidance backoffs, post-media arbitration timestamping, degree of FEC, retransmission and power management (cf. [**?**]). In addition to providing such control points, SP should expose costs (e.g., energy, delay, storage, bandwidth consumption) up to higher layers so protocols can optimize their behavior and how they exercise the available control.

For example, in addition to the traditional question of whether fragmentation occurs below or above the basic communication interface, when designing SP we must consider where protocol exchanges to reduce hidden terminals and improve fairness, such as RTS/CTS, belong. Exposing control of the underlying mechanisms allows middle layers to provide the additional functionality when it is needed at performance that is competitive to (or even exceeding) the general solution [?]. Simple defaults allow easy composition and use in basic operation.

Allowing multiple network protocols to share control over an underlying link layer raises concern as to how these protocols work together and cooperate. This is just the kind of investigation that the existence of SP would promote. We suggest that this question is tractable and very interesting in sensornets because they typically host a small number of widely distributed applications. In the Internet, Such control is problematic because the infrastructure is shared by arbitrary applications anywhere in the world. The classic operating systems boundary must address an arbitrary mix of complex applications. The application specific nature of sensornets is more conducive to cross-layer and cross-application customization.

In addition to an address-free broadcast, SP also has a directed send operation. The exact form this addressing scheme takes directed communication is an open research question. In the opposite direction of data flow, reception should provide an upward path for additional metadata and physical information, such as time of arrival, signal strength, and source. Additionally, reception demultiplexes packets across multiple network services and protocols.

4. FUNCTIONAL ARCHITECTURE

The functional components of a sensornet architecture develop around the SP abstraction. Here we briefly explore ramifications of SP design choices on these other aspects of the architecture. We illustrate how functionality divides among the various layers and investigate two tests of the architecture. First, we examine how the rich set of network layer protocols found in sensornets might be organized relative to SP. Then, we look at concerns that naturally cut across layers, using power management as an example, to see how SP might allow aspects of these services that naturally fall into different layers to cooperate without bypassing it.

4.1 Proposed Decomposition

Figure **??** shows a possible layer decomposition of a sensor network architecture. SP is the unifying abstraction that bridges network and application protocols to the underlying data link and physical layers. We anticipate multiple network layer protocols will reside over SP, with applications selecting specific ones. Highly specialized applications may embody their own protocols. We do not see a fixed naming mechanism playing a central role, as many of the most common protocols in sensornets eschew globally meaningful names, relying on proximity and physical connectivity to structure the flow of information. Based on current protocols and applications, we see two classes of multihop network layers over SP: address-free protocols and name-based protocols. As sensornets mature, time will tell whether more emerge or consolidation occurs.

4.2 Address-free and Name-based Protocols

Unlike an IP network, which supports a single network addressing scheme and provides mostly a single communication abstraction (*i.e.,* unicast), the sensornet applications developed so far use different naming schemes and require various communication abstractions. A name can have either local or global scope, and can be used to identify an individual node, a set of nodes, or a communication structure such as a tree. From the point of view of a node, a name represents a handler that can be used to process an incoming packet and decide on its next hop.

This variety is one of the main reasons behind our decision to push the narrow waist below the network layer. At the same time, the SP layer should present an interface general enough to support these various protocols. Trading between the requirements of these diverse network protocols and the desire to keep the interface as simple as possible is one of the main challenges of our architecture. To give a sense of what are the requirements imposed by these various network protocols and to illustrate their sheer diversity, in the remainder of this section we enumerate some of them.

Dissemination protocols and algorithms such as simple floods or Trickle [?] can use only the SP ability to perform local broadcast as an implicit naming scheme. This class of protocols is address-free: although they may include names to refer to data items – such as a sequence number – they do not use the concept of a node identifier. As transmit energy is a large concern, address-free protocols can elide destinations addresses when the data link layer allows it. [3]

Another important class of network-layer services provides multi-hop communication based on locally addressable nodes. An appropriate choice of routing protocol and associated addressing structure can enable more efficient communication. Protocols belonging to this class include data collection, dissemination, aggregation, and point-to-point routing. Key architectural issues that arise in designing these protocols include (1) what naming scheme to use, (2) how a node performs packet forwarding and processing relative to the naming scheme (data path), and (3) how a node discovers and maintains routing and processing state (control path).

Protocols such as converge-cast/collection routing, where nodes send data up a tree to a sink, are an interesting combination of the two schemes. On one hand, to a user of the protocol, it can be address- and name-free: the send is implicitly up the tree. The protocol is usually implemented, however, with next hop addressing to specify the path to the root.

In contrast to dissemination and converge-cast protocols, directed protocols such as geographic [?] or logical coordinate routing [?, ?] have a specified destination. More abstract and flexible naming schemes such as directed diffusion use data identifiers [?]. Global network names are powerful enough to support content-based storage within the sensor network, which requires any-to-any routing with low stretch [?].

Sensornet routing does not generally follow the Internet's end-to-end principle [?]. In addition to packet forwarding, a node along a path can inspect received data and make local decisions with it, possibly transforming the data before forwarding it, or suppressing it. This in-network processing can greatly reduce communication while keeping higher-level semantic requirements. For example, when collecting a MAX query over a network (which returns the maximum sensor reading), nodes need only forward the highest reading they receive, and can suppress if they hear higher readings.

4.3 Cross Layer Abstractions

One of the principal challenges to a sensor network is the defining the interfaces to services that cannot be effectively encapsulated in a single layer, such as power management, system management, timing, discovery, and security. Instead of being fully encapsulated at one layer, only visible to the one above and below, these services need to be accessible to all of the layers in the system. The design of SP has two challenges: providing an interface rich enough for application/SP collaboration, and keeping that interface platform independent.

As an example, consider a component of power management such as scheduling when the radio is off, in a low-power listening state, or fully

[3] For example, 802.15.4 radios have several addressing modes: 0-byte (address-free), 2-byte, or 8-byte.

active. The relative power costs and transition times between these states are platform dependent, but must be exposed in a platform independent way so protocols remain portable. Conversely, protocols must be able to provide the radio abstraction with information about their behavior, so it can manage the energy consumption of its underlying resources.

Each protocol and data source on a node may have its own communication frequencies and patterns: the node must be aware of all of them in order to best meet their needs. Encapsulating power management at a single layer only allows the layer above to provide information or specify behavior, but composition of many protocols requires a more flexible approach. For example, a simple collection application generates both data messages for a routing layer as well as control messages for the routing layer itself. Both the application and protocol need to be able to inform a node of their needs. Given these hints, a node can reduce its energy costs, by scheduling bursts of traffic. However, the interface needs to be richer than simple hints and queuing; the stack should inform components when they will be able to transmit, so those components can gather data accordingly.

SP must provide platform independent power management interfaces, but their details and tradeoffs are very platform dependent. SP must therefore provide a rich interface to protocols, informing them of the relative costs. Additionally, protocols need to be able to give hints or information to SP that allows it to make node or system-wide resource management decisions.

5. CONCLUSION

We contend that the main obstacle limiting progress in sensornet work is the lack of an architecture. A sensor network architecture would factor out the key functionalities required by applications and compose them in a coherent structure while allowing innovative technologies and applications to evolve independently. We argue that the narrow waist of this architecture should not be a network layer as in the current Internet, but a single-hop broadcast with a rich enough interface to allow multiple network protocols. This design decision is driven by the fact that, unlike an IP network, sensornets require a wide variety of naming schemes and communication abstractions.

However, there are many questions that need to be answered before such an architecture becomes a reality. Chief among those are the exact interface and functionality provided by the SP layer, and the interaction between SP and cross-layers such as power management.

6. REFERENCES

[1] B. Carpenter. Architectural Principles of the Internet. Request For Comments: 1958, June 1996.

[2] D. D. Clark. The design philosophy of the DARPA internet protocols. In *SIGCOMM*, pages 106–114, Stanford, CA, Aug. 1988. ACM.

[3] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler. Design of a Wireless Sensor Network Platform for Detecting Rare, Random, and Ephemeral Events. In *The Fourth International Conference on Information Processing in Sensor Networks (IPSN'05)*, 2005. To appear.

[4] R. Fonseca, S. Ratnasamy, J. Zhao, , C. T. Ee, D. Culler, S. Shenker, and I. Stoica. Beacon vector routing: Scalable point-to-point routing in wireless sensornets. In *Second USENIX/ACM Symposium on Network Systems Design and Implementation (NSDI)*, To appear.

[5] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister. System Architecture Directions for Networked Sensors. In *Architectural Support for Programming Languages and Operating Systems*, pages 93–104, 2000. TinyOS is available at http://webs.cs.berkeley.edu.

[6] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the International Conference on Mobile Computing and Networking*, Aug. 2000.

[7] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *International Conference on Mobile Computing and Networking (MobiCom 2000)*, pages 243–254, Boston, MA, USA, 2000.

[8] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A self-regulating algorithm for code maintenance and propagation in wireless sensor networks. In *First USENIX/ACM Symposium on Network Systems Design and Implementation (NSDI)*, 2004.

[9] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless Sensor Networks for Habitat Monitoring. In *Proceedings of the ACM International Workshop on Wireless Sensor Networks and Applications*, Sept. 2002.

[10] J. Newsome and D. Song. Gem: graph embedding for routing and data-centric storage in sensor networks without geographic information. In *Proceedings of the first international conference on Embedded networked sensor systems*, pages 76–88. ACM Press, 2003.

[11] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd ACM Conference on Embedded Network Sensor Systems*, 2004.

[12] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. Ght: a geographic hash table for data-centric storage. In *Proceedings of the first ACM international workshop on Wireless sensor networks and applications*, pages 78–87. ACM Press, 2002.