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*, Gilman Tolle*, Jerry Zhao†

1 Introduction

Wireless sensor networks have the potential to be tremendously beneficial to society. Embedded sensing will enable new scientific exploration, lead to better engineering, improve productivity, and enhance security. Research in sensor networks has made dramatic progress in the past decade, bringing these possibilities closer to reality. Hardware, particularly radio technology, is improving rapidly, leading to cheaper, faster, smaller, and longer-lasting nodes. Many systems challenges, such as robust multihop routing, effective power management, precise time synchronization, and efficient in-network query processing, have stable and compelling solutions. Several complete applications have been deployed that demonstrate all of these research accomplishments integrated into coherent systems, including some at relatively large scale [5, 17].

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 parts can be combined to build usable systems is quite limited. In order to produce running systems, 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 the work of others. This inherent incompatibility greatly reduces the synergy possible between research efforts and impedes progress.

It is the central tenet of this paper that the primary factor currently limiting research progress in sensornets today 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 promotes interoperability, transcends particular generations of technology and allows innovation.

2 The Nature of an Architecture

At the highest level, an architecture decomposes a problem domain into a set of services, which are functional components, their mechanisms and their responsibilities. An architecture can also define a set of interfaces to its services, which are the structures and functions services expose their mechanisms with. Finally, at the lowest level, an architecture can specify its protocols, which include packet formats, communication exchanges, and state machines.

For interfaces and protocols, we say an architecture can define them because sometimes it is advantageous not to. For example, the Internet architecture precisely defines IP as a service (end-to-end communication, best-effort delivery, fragmentation/defragmentation, etc.) and as a protocol (packet format and semantics), but is ambivalent to the IP interface. Given the Internet’s principal design goal — it is a network interconnection architecture — this ambivalence makes sense: IP does not want to dictate what software runs on each host.

In contrast to the Internet architecture, which seeks to promote communication interoperability, the POSIX architecture cares about software interoperability. Correspondingly, it cares greatly about interfaces while remaining ambivalent about the protocols. For example, the sockets service for end-to-end communication provides a precise interface, but has several underlying protocols (e.g., local communication, TCP, etc.).

The challenge in sensor networks is that their modes of operation introduce requirements and tradeoffs are very different from traditional systems. A sensor network application dictates sensor modalities and sample rates, real-time processing and data storage, and information exchange protocols among nodes. Early vision papers and analyses claimed that the traditional application/OS and network/data-link divisions are not well suited to sensor networks [7, 3]. Community experiences building protocols and applications have shown this claim to be true [27, 11, 16, 18]. Thus, current sensor network software systems, such as TinyOS [9] relax these divisions and give developers the flexibility to define new ones. This relaxation has allowed researchers to re-examine core issues in scheduling, power-control,
and information flow by cutting across traditional service boundaries [14].

Cutting across boundaries, however, has led to monolithic solutions or to subsystem components with arbitrary interface assumptions. While research groups have each been able to build large and complex systems, the resulting services, interfaces, and protocols are incompatible with each other. For future work to be able to build on the prior efforts of others, we need a sensor network architecture, which will re-establish a meaningful separation of concerns.

The Internet architecture demonstrated how a properly chosen set of guiding principles and services can shape the evolution of a complex system over vast changes in technology, scale, and usage [2]. 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 tremendous growth and change. However, this design is not free of costs; the use of rigid layering sacrifices 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 services, but in the overall ability to adapt. This is one of our goals 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.

The experiences and efforts of the sensor network community over the past years has helped shake out exactly how the requirements and concerns of a sensor network architecture are different from the Internet, and how they are the same. The challenge in defining a sensor network architecture is deciding what to specify in its services and what to leave open. Specifying too little will force systems to re-implement functionality they cannot depend on, while specifying too much will constrain future technologies and possibly lead developers to discard the architecture. For this reason, we expect developing an architecture to be at first a growing and organic process. While conclusions from community experience have clearly converged on some issues, such as packet timestamps, others, such as aggregation, are still under debate. By starting with services (or even parts of services) for which there is consensus, an architecture will help focus the research debate on open problems, promoting forward progress.

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 overriding question is whether there is a "narrow waist" — a functional component representing a common service 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 accommodates, even encourages, a vast degree of heterogeneity and diversity in both applications and underlying technologies. 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 devices and protocols to evolve and change without hampering optimizations. The Internet has shown that a narrow waist is the most important functional component of a network architecture.

We claim that sensor networks can also have a narrow waist, the Sensor-net Protocol (SP). Unlike IP, which is a multihop protocol intended for end hosts communicating over a shared routing infrastructure, SP is a single hop protocol. The reason for this difference is simple: sensor networks use a wide range of multihop protocols, such as dissemination [15], flooding, tree routing [26], and aggregation [18]. Applications differ dramatically in their communication patterns and are intimately tied to their associated network protocols. Most applications neither require nor benefit from a common, universally routable addressing scheme. Those that do can build such protocols on top of SP.

The first step in developing our architecture is defining the SP service by deciding which mechanisms and functionality it provides and which it does not. Using SP, protocol designers must be able to design a range of efficient 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 well defined service on top of a range of physical layers, how functionality divides across the packet boundary is a key question. To support the network protocols found in the sensornet literature, the mechanisms which a sender should be able to control include link level acknowledgments, post-media arbitration timestamping, retransmission and power manage-
Figure 1: Sensor Network Service Decomposition

4 Filling In the Architecture

SP is the keystone of our sensor network architecture, bridging higher level protocols and applications to underlying data link and physical layers. Defining the SP service requires understanding the requirements of applications that lie above it and the capabilities of the technologies that lie below. Just as with IP, it is unlikely that SP will be ideally suited to all of its possible uses. However, by examining applications and their requirements, we can make educated decisions on what trade-offs SP makes between its above and below pressures.

Applications today use a wide range of service layers, some of which have no clear analogues in the OSI model. For example, several commonly used communication services, such as collection routing [26] and dissemination [15], are address-free, in that, from the perspective from an application, there is no explicit destination. Of course, there are also name-based communication services, but the form and semantics of the naming are very different than end-to-end communication.

Address-free and name-based communication represent traditional service layers, which encapsulate underlying functionality. Our sensor network architecture also has cross-layer services, which cut across SP and indeed the entire architecture. Deployed applications have demonstrated that there are pieces of information and functionality which many different services require concurrently. Establishing the concept of cross-layer services allows existing approaches to continue while providing the structure necessary to promote compositability and reuse.

4.1 Proposed Decomposition

Figure 1 shows a possible decomposition of a sensor network architecture. SP is the unifying service that bridges protocols and applications to the underlying data link and physical layers. Situated above SP are multiple network layer services, with applications selecting specific ones that suit the networking needs of the application. In Section 4.2, we discuss two of these upper layer services, name-based and address-free protocols. Situated below SP are underlying data link and physical layers, such as 802.15.4 [25] or S-MAC [28]. The diversity of functionality underlying layers present poses a variety of technical challenges to SP’s design, which we discuss and address in our SP proposal [21].
In addition to the layered services above and below SP, the architecture has cross-layer services, which Figure 1 shows on the left side. Cross-layer services include power management, timestamping/time synchronization, and discovery. As we discuss further in Section 4.2, these services are cross-layer in that they have uses across the entire spectrum of service layers.

This decomposition is far from complete. As sensor networks evolve and spread into new application domains, it is inevitable that new services will emerge. Current and foreseen future uses motivate our current decomposition, but it is also intended to be flexible enough to engender growth.

4.2 Address-free and Name-based

Unlike an IP network, which supports a single network addressing scheme and largely provides a single communication abstraction (i.e., unicast), applications developed so far use a variety of naming schemes and multihop communication services. For example, LiTES [5] routed along a 2D grid, GDI routed up a collection tree [17] and PEG used a landmark routing overlay on top of a tree-building algorithm [23].

This variety in naming and multihop communication is one of the main reasons behind our decision to push the narrow waist below the OSI network layer. Lowering the narrow waist allows the architecture to express and encompass this diversity both in the present and in the future. Trading off between the requirements of higher level services and the desire to keep SP as simple as possible is the principal first challenge in developing the architecture. In the remainder of this section, we describe two higher level services and how they might influence SP. Key architectural issues that arise in designing these services include route discovery and maintenance, naming, and the packet forwarding rules.

The address-free service layer encompasses a wide range of protocols, including flooding, collection routing [25], dissemination [15], and aggregation [4]. Although these protocols may include names to refer to data items — such as sequence numbers or dispatch IDs — they do not identify nodes directly. For example, when an application wants to send a piece of data up a collection tree, it does not need to specify a destination because it is implicit: the node’s parent in the tree. The underlying collection tree routing protocol may address the parent directly, but it encapsulates this naming and hides it from layers above.

Unlike collection routing, however, which typically names nodes at the SP level, broadcast and dissemination protocols rely on the implicit naming provided by local connectivity. This represents an interesting SP design consideration, as some underlying MAC layers (e.g., TDMA-based MACs that turn off the radio) may not by themselves provide an efficient local broadcast primitive. This tension between the requirements of layers above and the capabilities of layers below demonstrates some of the difficulties that designing SP presents.

The name-based service layer encompasses multihop communication based on destination identifiers. This includes approaches such as geographic routing [12] and logical coordinate routing [19, 8], as well as more abstract and flexible naming schemes such as directed diffusion, which use data identifiers [11]. Global network names are powerful enough to support content-based storage within the sensor network, but require any-to-any routing [22].

In addition to packet forwarding, a node along a path can inspect received data and make local decisions regarding a packet based on its contents, possibly transforming the data before forwarding it, or suppressing it completely. This in-network processing can reduce communication while keeping higher-level semantic requirements. For example, when collecting a MAX query, which returns the maximum value of some variable, nodes need only forward the highest value they receive and suppress all other values.

The key observation is that the services above SP support very different semantics than those found in the network layer services of the Internet and OSI specifications. In particular, sensornets are primarily concerned with dissemination, collection, aggregation, and gradient-directed services, whereas the Internet is principally concerned with end-to-end communication [1].

4.3 Cross-Layer Services

One novel aspect of our sensor network architecture is the concept of cross-layer services. Cross-layer services cut across layers or arise within multiple layers. Instead of being fully encapsulated at one layer, only visible to the layers above and below, these services are accessible to all of the layers in the system. In this section, we use power management to motivate why cross-layer services are necessary in a sensor network architecture, describe some of the research challenges they pose, and present timestamping as one example.

Energy constraints are a defining characteristic of sensor networks. Traditionally, power aware networking has been dealt with at a single point in the stack in isolation. This approach is not practical in sensornets because power management often appears in many places and takes many forms. Below SP, power aware MACs attempt to turn off the radio invisibly to the stack above [24]. Within SP, buffering multiple packets and sending them back-to-back in a burst can be more efficient than sending them individually as they appear from the network layer services [21]. Routing layers above SP
can have multihop flow information that allows them to schedule future radio activity [10]. Applications can also have scheduling policies: TinyDB shuts down the whole networking stack between query processing epochs [16].

A consequence of its ubiquity is that power management is particularly challenging to abstract into a clean architectural concept. The architecture must allow many different services from very different levels to collaborate and work together. These services must therefore be accessible to all levels of the system. On one hand, these services must have policies to arbitrate between conflicting requests; on the other, constraining the possible policies unnecessarily will hamper future growth. An architecture must establish clear guiding principles and sufficiently rich, yet loosely-coupled and appropriately abstracted, interfaces to support cross layer services.

All of the power management approaches mentioned above use some form of time synchronization to schedule communication. All of these time synchronization algorithms depend on having accurate packet timestamps. Therefore, these timestamps must be information that cuts through layers so sub-SP as well as super-SP services can use them. While time synchronization services can be situated above SP, MAC-layer timestamping below SP greatly improves precision [13]. By choosing an to generate this data at an idealized point in the communication stack (post media arbitration) SP can achieve microsecond resolution inexpensively.

The point is that timestamps must cross layers so the architecture should provide abstractions to access and manipulate these timestamps. The preferred method of exposing timestamps in the link interface, and more generally across the architecture, is an important design point that must be addressed with an eye toward removing any temptation for time coordination services to circumvent SP. Power management is an example of a cross-layer service for downward control; timestamps are an example of a cross-layer service for upward information flow.

While this section presented only two cross layer abstractions, there are many more that we need to address. Examples include system management, discovery, and security. These services need to be accessible to all of the layers in the system so their abstractions present a central challenge to the architecture’s design: developing a methodology for providing interfaces rich enough for application/system collaboration while remaining flexible enough to encompass growth and evolve as time rolls forward.

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 services 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 single-hop communication with a rich enough interface to allow a diverse range of network protocols. This design decision is driven by the fact that, unlike an IP network, sensornets require a wide variety of naming schemes and multihop communication services.

However, there are many questions that need to be answered before such an architecture becomes a reality. Chief among those are the functionality provided by the SP service, the functional decomposition of sensor networking into services now that the narrow waist is single-hop, and how cross-layers services such as timestamping can be designed to enable a broad spectrum of uses while minimizing complexity. Our hope and goal is that such an architecture will enable research groups to more easily collaborate and build on each other’s efforts. Rather than a set of incompatible and vertically integrated systems, we will in the near future see in sensor networks the variety and innovation we see in the Internet today.

Acknowledgements

This work was supported by the National Science Foundation, under grant CNS-043545.

References


