

Network Stack Architecture for Future Sensors

Rajnish Kumar[†], Santashil PalChaudhuri[‡], David Johnson[‡], Umakishore Ramachandran[†]

[†] College of Computing, Georgia Institute of Technology, Atlanta, GA,

[‡] Department of Computer Science, Rice University, Houston, TX.

Abstract—With wireless ad hoc sensor networks, there is simultaneously a need and an opportunity to optimize the protocol stack behavior to match the sensor-based applications. A general-purpose internet stack is neither appropriate nor sufficient to meet the needs of such applications. Motivated by this observation, we pose two related questions: (a) What is an appropriate layering of the protocol stack for future sensor networks? (b) How do we make the network stack tunable to a specific sensor application? We present the design of a new protocol stack, and we qualitatively argue that the new stack is more suitable to meet the demands of sensor network applications than traditional stacks.

I. INTRODUCTION

Wireless Ad hoc Sensor Network(WASN) [25] has recently attracted a considerable amount of attention due to its application potentials. Despite similarity to research issues in ad hoc networking and distributed systems, there are several requirements that are fairly unique to WASN leading to many recent innovations to the different layers of the traditional protocol stack. These unique characteristics of sensor networks are as follows:

- *Data-centric routing*: The data centric routing need arises from the fact that the specific data sources may be unknown for some applications. For example, an application wishes to find out the maximum temperature in a region, and so the sensor nodes to be contacted for this purpose is not known a priori. To handle such needs, low-level naming of the nodes and interest-based data routing have been proposed by recent research [11].
- *Energy constraint*: There are many applications where WASN will be deployed in remote areas, such that the nodes are either physically unreachable, or the cost of recharging them prohibitive. To increase the WASN application lifetime, there is a need to optimize energy consumption at all layers of the network stack in every possible way.
- *Data Fusion*: Data fusion can either be an application-specific need [19], or it can be the property of the sensed data, such that there exists some correlation in the sensed data, and the data collected from different sources can be fused together en route to the sink. The in-network data fusion has been a very useful technique for saving communication

cost. Support for this fusion needs to be inbuilt into the stack for performance reasons.

- *Adaptability*: Sensor networks have a limited set of applications with specific set of requirements. So, the stack can be optimized for these applications, which is not possible in completely generalized application scenario.
- *Large scale*: WASN is expected to consist of thousands of sensor nodes, much larger than typically realized in ad hoc networks. This makes the traditional way of stateful routing impractical because of the size and number of routes.

Traditionally, layering has been used as a design principle for networking stacks. This technique organized a network system into a succession of logically distinct entities, such that the service provided by one entity is solely based on the service provided by the lower level entity. This is important to deal with complex systems, as the explicit structure allows identification of the relationship between the pieces. This modularization eases maintenance, updating of the system components, and the change of the protocol of a layers service is transparent to the rest of the system. We argue in this paper that the traditional protocol stacks are not appropriate for WASNs.

End-to-end Vs Hop-to-hop Data Fusion: While TCP/IP stack is well-suited for end-to-end requirements, it is not appropriate for hop-to-hop requirements of WASN. This is because, many sensor applications need data packets to be considered for data fusion that is typically done at the application layer. If TCP/IP is naively changed to allow the data packets to reach the application layer at every hop, it will lead to performance penalties. Since the data packet may have to travel large number of hops before it reaches the sink node, it will incur large end-to-end latency.

Irrelevance of some TCP/IP stack services: Some properties of TCP/IP stack include flow control at the transport layer, node fairness at the MAC layer, and error control at *both* the MAC and transport layers. While some of these services may be required by selected sensor applications, they are not generic requirements for all sensor networks applications. Hence putting them in the stack lead to considerable performance penalty for applications not needing them.

New service requirements: There are some services that are considered trivial for TCP/IP stack, but become important and non-trivial to support for WASN environment. Two such services are: *Node addressing*, *Location Awareness*, and *time synchronization*. Attribute-based naming has been found useful for providing the flexibility demanded by WASN [29]. This naming mechanism is different from IP-based addressing, and affects the strategy used for routing. Current TCP/IP stack is suitable for IP-based addressing, but it needs to be adapted to support this logical naming. Sensor applications typically require location awareness to satisfy localized queries. This awareness also enables routing and MAC protocols to optimize on efficiency and latency. Time synchronization is an important functionality needed by all layers for cooperation among sensor nodes. The OSI reference model or the TCP/IP stack either leave the onus to applications or implicitly assume presence of time synchronization.

Energy optimization at every layer: Energy is a very important resource for sensor networks because it determines the application lifetime. Ideally, the energy usage can be maximized if the whole stack is integrated and customized for a specific application. With the traditional protocol stack, it will be difficult to do any energy optimization in an integrated fashion, or to adapt the protocol behavior according to the application needs.

In an effort to address these WASN characteristics, most research efforts to date in sensor networks have either completely ignored the traditional network stack and specialized the transport to suit the specific approach (e.g. data diffusion [15]), or have made localized modifications using the traditional *Open Systems Interconnection* (OSI) reference model [16] as the de facto standard (e.g. PicoNode [26]). The standard sensor node OS called TinyOS [12], provides the lowest layer of networking stack, MAC, giving a messaging service, and all routing and fusion logic is provided as an application on top of it. This is done due to the limited resources of the current sensor nodes. Our proposed stack extends this to provide services like fusion, routing, filtering, and synchronization within the stack in more powerful future sensor nodes. There has also been considerable research effort directed at the different layers of the OSI model, but most of them have tended to treat the layers independently as encouraged by the OSI model. Though there has been some cross-layer optimization efforts, much more optimization is possible if the layers are treated together.

We believe that this is an appropriate time to think of a new design of the protocol stack for future sensors. We also believe that CPU and memory will not be constraining resources in future, but energy will remain to be. As sensor networks will become pervasive and powerful in the foreseeable future, it is counter-productive to design a customized stack for each application. On the

other hand the traditional stack is inappropriate to WASN as explained above. With this motivation, in the rest of the paper we list the design goals of a stack for sensor networks and explain our architecture.

II. WASN STACK DESIGN REQUIREMENTS

This section identifies a set of requirements for the SensorStack, a new protocol stack suitable for WASN environment.

In-stack data fusion : SensorStack should provide mechanism to do data fusion in the protocol stack. If fusion logic is made part of the protocol stack, the data packets do not need to reach the application layer at the intermediate hops between source and destination nodes. This will improve the end to end latency of data packets. Also, providing the common fusion requirement of sensor application as a separate module, application development can be made more modular. Thus, same fusion code can be used for multiple applications alleviating the load on resource constrained sensor nodes.

Logical naming support : SensorStack should understand logical naming, and it should provide mechanisms to utilize the logical naming.

For example, if a node wants to send a data packet to gather temperature from the nodes in region X , the logical name of the destination becomes : “nodes in region X ”. This can be supported by flooding the information, as is done in directed diffusion. Here, every node who listens to the packet will compare its own location with X , and accordingly it will decide whether to reply with its temperature data or ignore the packet. This happens at application layer, which means the data packet has to travel to the application layer at every node in the network even for nodes not lying in the region X .

For now, assume that the protocol stack at a node has access to the location information of the node. Can we use this location information to stop packets traveling to upper layers when it is not required? This can be achieved if SensorStack can understand the notion of logical naming, and thus it can filter the data packets at network layer itself (at *non-interesting nodes*) and rebroadcast the packet for attention of neighboring nodes. This idea is similar to MAC-level filtering done in bridges [24].

Adaptable to a specific application : SensorStack should allow applications to tune the stack behavior and thus adapt itself to application-specific needs. Applications will have access to information about the user requirements, the network topology, etc, and these information can be used for improved decisions at the protocol stack : which route to take for data transmission, whether to do error checking or not, what duty cycle should be used for the low-power radio, etc.

Supporting application-specific adaptation can be taken in two extremes: *first*, stack is implemented for a par-

ticular application and the application directs the stack behavior without any intermediate abstractions, and *second*, the stack is implemented for a class of applications and applications direct the stack behavior via a broker service. SensorStack adaptation is not suitable for the first category, because this will need different protocols to be written for different sensor applications and these protocol codes to be loaded into network stack at run-time. Since the class of sensor applications have many common requirements, it seems plausible to support the application-specific adaptation via the broker service.

Cross-layering : SensorStack should make the relevant information from one layer available to other layers such that the other layers can take more informed decision. The resource constraints of WASN demands this to achieve optimization in an integrated way.

III. NEW PROTOCOL STACK DESIGN

Based on the requirements discussed in previous section, here we present our proposed SensorStack.

Our proposed stack is based upon following observations that span across all WASN requirements :

- Bring WASN specific service needs, data fusion and logical naming, in the stack,
- Remove those services from the stack which are either irrelevant for WASN or can be better supported at application layer, e.g. transport logic, and,
- Add a broker service that allows different services in the stack to talk to each other and the application in a standard way.

Figure 1 presents this new proposed stack. The three main mandatory services needed to the sensor applications are medium access control (MAC), logical naming and data-centric routing (data service), and data fusion. These three services are in three separate layers. Other services that can improve the performance and energy optimizations, e.g. localization service, are put together as optional service bundle. Information exchange service (IES) is acting as the broker among different service modules and the application to improve cross-layer optimizations.

To support data-centric routing, SensorStack addresses nodes with their logical names, i.e. a set of attributes and values. If we consider IP-address as yet another attribute, then address-centric routing becomes just a special case of data-centric routing. Based upon this observation, SensorStack treats IP-address as a logical name, and it allows application-specific ways of logical name interpretation for data routing decisions.

Below, we describe the SensorStack services in detail.

Medium Access Control Layer: This layer provides the traditional TCP/IP's MAC layer service, i.e. medium access for hop-to-hop data transfer and the channel error control, and the power control of the radio antenna. The

sensor network MAC should be very energy efficient and scalable.

Since probability of channel error is higher with wireless compared to wired medium, error control is an important service to be provided by the MAC layer whereby application might specify the degree of reliability it needs from the MAC. The MAC layer also controls its own duty cycle, and the wake-up and sleep schedule of the radio. It can use the IES service to optimize the energy efficiency of medium access and to control the duty cycle, e.g. IES can provide the location information of neighbors, or application data communication schedule. Also through IES, the communication requirements of the application are published, which is utilized by the MAC for better energy-efficiency and lower latency.

Data Service Layer: This layer provides two main services, namely, dissemination of (potentially fused) data to one or more neighbors, and reception of data packets for fusion or delivery to the application. To support these two services, this layer needs to implement the following functionalities:

- 1) Packet Scatter/Gather: If the message size is greater than the MAC layer packet size, then a message may need to be fragmented at the source, and reassembled at both intermediate fusion points as well as ultimate destination points.
- 2) Logical naming and filtering: This functionality resolves the logical naming of data packet addresses. This is called only for the incoming data packets. Data service layer uses the information available from IES to resolve the logical name, i.e. it matches IES provided values for the packet attributes with those available in the packet itself, and it forwards the packet, for fusion consideration, to data service layer in case of a match. In case of a mismatch, the data packet is passed, for data forwarding consideration, to next hop selection functionality.
- 3) Next Hop Selection: This function takes the data routing decisions. Based upon the destination's logical name, it finds out the next hop's logical name and updates the data packet header.

Data Fusion Layer: This layer needs to support both types of fusion mechanisms: first, where the data packets are required to be fused at every hop, second, where the data packets are to be considered for fusion only at the (application-specified) selected nodes or the destination before the data is delivered to the application. The first kind of fusion mechanism is meant for hop-to-hop data transfer, and is useful for application scenarios where every node is sensing some useful data that needs the in-network fusion [20]. The second kind is useful for supporting more traditional way of doing fusion at the end-points, as sometimes required when only some nodes are contributing towards the information that is being

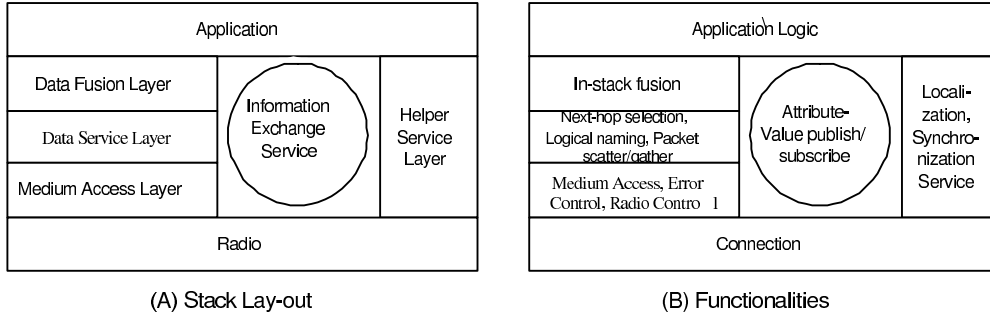


Fig. 1. Proposed stack for sensor networks. Left diagram shows the top-level modules and their relative boundaries. Right diagram lists out the functionalities provided by the modules in left diagram.

sought [19].

SensorStack provides hooks and mechanisms for safe execution of application-supplied fusion function in the kernel. It uses *fusion channel* abstraction, similar to that in DFuse [19], to capture the fusion semantics. A fusion channel can be thought as a black-box that takes a set of input streams, applies the fusion function, and generates an output stream. The application creates a fusion channel by making following call :

```
createFusionChannel(handler, exceptionHandler,
                    argInput1, ... argInputK, argOutput)
```

In above call, application specifies the handler fusion function, a exception handler function, and a set of arguments. Each argument is a tuple of data type and data source. Last argument, *argOutput*, is meant for the output data item. Data fusion layer maintains a set of data queues, one queue for every data type. Thus, there will be $k + 1$ buffer queues for a fusion channel with k input data streams. Data service layer enques data packets to these queues based upon data type field. When data item for each of the input queue is available, fusion function handler is invoked. Exception handler is called when the input data items are not available within a prespecified timeframe. The output data item of the fusion function is wrapped with information from *argOutput*, and enqueued in the output buffer queue.

Allowing fusion functions to run in kernel-space needs sand-boxing and special limitations on the programming for the safety reasons [1]. SensorStack uses segmentation and sandboxing based techniques for the safety reasons. It also maintains strict bound on the input and output buffer queues, and the number of handler calls pending at a time for any fusion channel.

Information Exchange Service: This service acts as a information database. It serves two main purposes: allowing cross-layer optimization by making one service’s information available to another service, and making application requirements available to the different layers. Potentially, a service could directly invoke another service to get the information it wants, but that will lead to two issues: a service will need to know which service to

call for to get the information, and very often kernel-level services will need to make user-space calls to get application-specific information. By having a generic information exchange service, SensorStack can make the information available transparently, and it can control the information access/update rights in a centralized manner.

This service can be provided as a publish/subscribe API. Different protocol layers and the application layer can publish/subscribe the relevant information. For example, network monitoring service may be running at the application layer and it can publish the health information to IES. This health information can be subscribed by different protocols in the stack. It can be used to do the next hop selection Similarly, location information can be used to support geography-aware routing. It can also be used by MAC layer to do energy optimization, like adjusting the transmission power or to adjust the directional antenna.

Apart from location, other information that we believe will be useful if provided by the IES are:

- Time: Time synchronization is getting more essential for WASN as it is getting useful to support energy-efficient MAC protocols, or to support time-sensitive application functionalities.
- Data communication schedule: This information can be used to schedule the radio wake-up and sleep, or to do energy-efficient medium access control. Application can directly publish this information if it knows its communication need, or a sub-service can be implemented that generates the communication schedule by looking at the traffic.
- Data packet size: The data service layer and the MAC layer can use this information to decrease the packet fragmentation.
- Latency: The MAC layer can provide the latency between itself and a neighboring node, and this information can be useful to do congestion control by the data service layer.

Application and Helper Service Layer : Application layer is responsible to support data capturing, data presentation at the sink, instantiating fusion channel at ap-

appropriate nodes, and providing other services that can be used to adapt the SensorStack for the application-specific demand. The other services can be executed either in user space, or it can be run in kernel-space (in sandboxed manner similar to fusion handler) in helper service layer. We think that localization and synchronization are two important services that need to be part of SensorStack, and we have placed them in the helper service layer. The localization and synchronization service publishes location and time information to the IES.

IV. IMPLEMENTATION

A. IES:

- namespace - same namespace
- how can a subscriber specify the fidelity of information it wants ? - by adding corresponding attributes in the IES
- size of IES - limit the number of entries a particular service can insert
- where is IES located - kernel heap
- how to restrict the publisher - no restriction, only the number of entries one can enter
- a service writer needs to be aware of the namespace convention (including name of attribute, how to interpret the supplied value, and which subscriber it should trust)
- true, but the limitations are similar to that of pre-defining the interfaces !
- advantages: central approach to safety - controlling who can write (including application)
- application can publish, and the kernel-level services need not make upcalls
- expiration period of a published attribute-value
- mechanism : in beginning, every service populates the IES with the attributes it CAN publish. after that, when ever any service needs IES information, it will

B. Fusion Layer

- function invoked as soon as the data items are available in the buffer queues
- rate of fusion needs to be controlled inside the fusion code
- Source of every input item is also specified
- the logical name representation of the encoded source name (in packet) may be different from the one using which fusion channel was specified ???
- argOutput
- No timestamping
- fusion channel migration
- restrictions on the fusion function implementation:
- bounded execution time ?
- restriction on the system calls it can make - createFusionChannel(handler, exceptionHandler, inputArg1, inputArg2,...inputArgK, outputArg) where, inputArg = < data type, data source, #items >

C. Application Layer:

- needs to create the fusion channels
- needs to support the IES interface - publishableAttrList(), and getVal(attr)
- need to provide data-type/protocol mapping if any new data type is being handled

D. Data structures:

- fusion data queue
- IES table
- Fusion function list
- data type - protocol mapping

V. APPLICATION SPECIFIC STACK ADAPTATION

In this section, we explore the design space for the adaptation and optimization of our protocol stack for two particular application scenarios.

A. Application Scenario

Consider the case of monitoring a large area for temperature sensitive events, like intrusion of a mobile object (e.g. tank), or an event of fire. The WASN consists of scattered sensors and cameras. The sensors are needed to sense the temperature information, while the cameras are used to take the images in case of a fire or intrusion detected in the area. The user, sitting at the sink node, monitors the temperature statistics of the area that is presented at regular interval. The user, as a sanity check could request the WASN to report an image-scan, a composite of several images from different regions, that is generated by fusing together the image data from the different cameras, en route to the destination from the source nodes, by applying some predefined correlation function over the scanned images.

The application level support for realizing the above scenario is well-studied by recently proposed frameworks: TAG [20] can be used to provide the temperature statistics in an energy-efficient way, and DFuse [19] can be used to present the correlated image scans to the user in the WASN environment. The main point to take away here is that *the fusion and transport level needs for the two types of data, namely, temperature readings and images, are completely different*, and so neither TAG nor DFuse is suitable for supporting the complete scenario by itself. The sensed data needs to be aggregated at every node from the source to the sink, while the scan image data needs to be correlated only at selected nodes. Next, we briefly discuss the two frameworks, TAG and DFuse, to understand their network level needs.

TAG is a generic fusion service for sensor networks. There are two essential attributes of this service. First, it provides a simple, declarative interface for data collection and fusion. Second, it intelligently distributes and

executes fusion queries in the sensor network in a time and power-efficient manner. The two basic types of query that are supported in this framework are *periodic* that specify the interval a sensor should generate and aggregate data to send to sink, and *event-driven* that registers interest in a specific event a sensor should monitor and send notification when it happens. To support the above described application scenario, the *periodic* query scheme can be used to collect the temperature statistics at regular intervals, and the *event-driven* query can be applied to detect the intrusion or the fire.

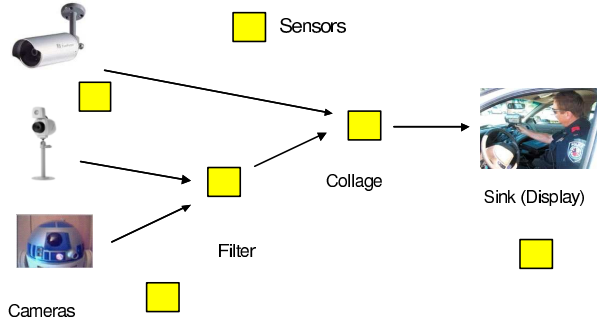


Fig. 2. An application task graph for generating image-scan.

DFuse is a framework for supporting hierarchical data fusion in WASN environment. It takes the application task graph and the fusion codes as the input, and deploys the task graph over the network such that the overall transmission cost is minimized. For the above application scenario, a possible task graph can be as shown in Figure 2. While the data sources and the data sink in the task graph are application specified, the fusion nodes are selected by the DFuse framework.

B. Application and Data Fusion Layer

Application layer now is a thinner layer compared to the case of TAG/DFuse running on top of TinyOS or IP-Stack. DFuse’s placement algorithm, responsible for fusion point placement, still runs at the application layer. Also, data capturing at the source nodes and data presentation at the sink node is done at the application layer.

Fusion logic of TAG and DFuse, earlier supported in application layer, will run in the data fusion layer of the SensorStack. The application layer is responsible for creating the fusion channels at the network nodes, thus providing the fusion code, the destination nodes names, and input data types and their sources. TAG packets and DFuse packets are differentiated by the data type attribute.

C. Data Service Layer

The data routing and naming needs for TAG and DFuse are quite different as explained in the application scenario.

Supporting both the hop-by-hop and end-to-end routing needs of TAG and DFuse is difficult in a traditional stack. On the other hand, the data service layer of the new stack can use logical naming to support hop-by-hop routing, while the physical naming with logic for location-aware next-hop selection can support end-to-end routing.

TAG uses a simple routing scheme based upon the tree structure rooted at the sink. For the purpose of tree-based routing like TAG, the IES will need to provide the parent information for a given node. Thus, next hop for every TAG packet coming from the source nodes is the parent. Thus the hop-by-hop routing for TAG can be achieved by logically naming the destination as the current node’s “parent”.

DFuse data routing can be served by any location-aware routing algorithm, such as SPEED [9], or GPSR [17]. The main advantage of using the new stack lies in the fact that these algorithms can improve the quality of their next hop selection by utilizing the IES information. Using cross-layer information to optimize the data routing decision has been proposed by recent research projects [9], [10]. SPEED [9] uses MAC layer delay estimation for doing the next hop selection. GEAR [10] uses location-information to optimize the data flooding in directed diffusion. Diffusion algorithms propose to use application-specific filters to optimize the distribution of interests [15]. These cross-layer optimizations come at the cost of trading off the modularity of the traditional stack. The proposed stack allows supporting these optimizations without losing the modularity by providing standard publish/subscribe interface for information exchange.

The new stack allows applications as well to adapt the routing behavior to suit their own purposes. As an example, consider tuning directed diffusion routing algorithm such that nodes in a particular region are considered important and that their power needs to be conserved compared to nodes in other regions. Directed diffusion uses latency as one of the factors to reinforce a preferred neighbor. In the new approach, the directed diffusion can use application-guided ranking of neighboring nodes to reinforce them. The IES, having access to the location and application’s topology information, can help the diffusion algorithm do the neighbor ranking.

D. Medium Access Control Layer

In shared medium networks, one of the fundamental tasks of a MAC layer is to avoid collisions between two interfering nodes. It allocates the channel to the nodes efficiently, so that each node can communicate with a bounded waiting time and with as little overhead as possible. The important attributes for traditional MAC are fairness, latency, throughput and bandwidth utilization. In contrast, the important attributes of a MAC protocol for WASN are energy efficiency and scalability towards

size and topology change. The major sources of energy wastage as elaborated in [30] are:

- *Collision*: Collision results in corruption of a packet and subsequent retransmission leading to increased energy consumption as well as latency.
- *Idle listening & overhearing*: Listening for either possible packets or packets destined for other nodes leads to wastage of energy. Idle listening consumes significant energy comparable to actually receiving a packet.
- *Control packets overhead*: Increased control overhead leads to increased energy usage in direct proportion.

Though the proposed MAC protocols [27], [28], [30] for WASN have identified and addressed many of the WASN *environment* requirements, they have not taken advantage of the nature of WASN *applications*. With the availability of the IES in the proposed stack, following properties of WASN applications can be exploited for the MAC protocol design:

- Communication requirements may be *periodic* and known beforehand such as collecting temperature statistics at regular intervals. This information can be used to schedule the medium access by the nodes and thus minimize collisions, as well as aid the radio interfaces sleep/wake-up decisions thus decreasing the idle listening and overhearing. Also, if every node knows what data packet it should receive periodically, then a NACK based medium access can decrease the control packets overhead.
- A contention based medium access will also be necessary to support *event-driven* applications like intrusion or fire detection. With information from the IES, the forwarding node can be woken up in time to process event-driven data. Real-time constraints can be communicated from the application to adapt the MAC to meet its requirements.
- Often the sensed data packet may have fixed size. This simple information will help minimize doing packet fragmentation and assembly at MAC layer, thus improving the throughput [18].

An application-adaptable energy-aware MAC protocol can be build upon the previous sensor MAC protocols, that exploits the known application requirement via the IES. The MAC protocol should have two modes to support the two different communication requirements of sensor applications, namely periodic and event-driven. The need to support these two kinds of modes was recently proposed in the IEEE standard for low-power sensors [14]. The relative proportion of the two modes in a superframe structure is dynamically determined by the applications depending on their current needs.

Periodic Contention Free Period: Medium access in this mode is based upon Spatial Time Division Multiple Access (STDMA) [21]. The application's deterministic

traffic distribution during the periodic communication can be used to compute an efficient slot allocation policy similar to [3], [8]. The length of the slot is enough to send a complete data packet of fixed size, allowing TAG like applications to send periodic sensor readings. The forwarding node expects data from a node periodically, so there is no need to waste energy to send an explicit ACK, and it sends a NACK when it does not receive the expected data. Using the known traffic pattern, each sensor will be in active mode only when communicating, and in sleep mode for the rest of the time, leading to a whole lot of energy savings because of this low duty-cycle.

Event-Driven Contention Access Period: During this mode, a sensor will be in sleep-mode except when necessary to communicate. For sending event-driven data like the image scan of a specific location, this mode of communication is used to send the data to the next fusion node. The mode is based on IEEE 802.11 protocol [13] with carrier sense and RTS-CTS. Techniques such as overhearing of neighbor's NAV vector citepamas to save energy, and sacrificing per-node fairness for lesser collision [30] is Sometimes fragmentation is unavoidable but fragments of data is not much useful for sensor nodes sensors receive all the data and then do the processing. So, the whole Application Data Unit(ADU) is sent together and assimilated at the receiving node [4], thereby decreasing the number of collisions and control packet overhead and hence energy requirements.

Apart from using the deterministic traffic of WASN applications for efficient medium access, the MAC uses other IES supplied information like the sensed data size(s) to determine the slot duration and back-off time. Using location information, the MAC controls the transmission power to the lowest level necessary to reach the next hop, as well as gleans information about the direction of the next hop for use in directional antenna. Various real-time guarantees sought by the application is communicated via IES to configure the MAC to meet those constraints, thereby adapting it to serve the application in the most efficient manner.

E. Adaptive Network Services:

Clock synchronization and localization are very important services in the sensor network systems. In our SensorStack we provide interfaces such that the accuracy of the service provided depends on the need of the application and the current resource constraints in the sensor nodes. The applications and rest of the stack interact with the IES to provide the service requirements, and the service protocols provide those set of requirements.

For clock synchronization, we have proposed an adaptive method for clock synchronization [22] that is based on the Reference Broadcast Synchronization [5]. During the start of each superframe, each node which is a forwarding node sends out a reference beacon using which

all the nodes in hearing range synchronize with each other. We provide a probabilistic bound on the accuracy of the clock synchronization, allowing for a trade off between accuracy and resource requirement. Expressions are derived to convert these service specifications (maximum clock synchronization error and confidence probability) to actual protocol parameters (number of messages and synchronization interval).

VI. DISCUSSION

Here we touch upon some of the specific aspects of the proposed stack which have not been explained earlier.

Sensor applications have varying degrees of security concerns. Traditionally, the data integrity is supported at the physical layer in wireless networks, and data authentication is supported at higher layers. WASN applications will need the authentication support below the data fusion layer, because unauthenticated data packets must be discarded. Data service layer can support the data authentication service. IES can help support this requirement without any change in the layering of the proposed stack. It can make the application security keys available to the data service layer.

Though current WASN applications do not need the full functionality associated with the transport layer, future applications using sensor networks may need it. We expect that the transport layer functionalities can be supported by middleware running over the proposed stack, and the middleware can host the future sophisticated sensing applications.

WASN may need to co-exist with other types of networks. For example, users may want to access the sensed information via the Internet. The interface nodes will have to support both the proposed network stack and the stack corresponding to other networks.

VII. RELATED WORK

OSI reference model provides the basic framework for development of standards for interconnecting two or more systems. TCP/IP stack has similar motivation and structure as the OSI model. Each layer in OSI provides a set of *services* to the subsystems in the layer above. In providing these services, a layer implements a set of *functionalities* using the services made available by the layer(s) below.

The layered model of the TCP/IP stack and the OSI reference model encapsulate the issues appropriate to the related tasks within each layer. In a standardized way, this layering allows transparent access to all lower-level functions, and makes it possible to upgrade any given layer without the redesign of other layers. The strength of layered approach lies in its modularity. Every layer now can be supported by different vendors, or implemented for different hardware platforms, and yet the overall stack can be realized by simply combining the appropriate protocols

for every layer. This modularity leads to simplicity by hiding the complexities of the lower layers.

While the TCP/IP stack has been in common use on general purpose machines, they have also been adapted for more specialized networks or application needs. Specifically, research projects have looked at the scope of cross-layer optimizations in TCP or UDP stack for wireless networks. For example, there have been proposals [23], [9], [31] for sharing the physical and MAC layer knowledge of wireless medium with the higher layers for efficient allocation of network resources or for congestion detection. Similarly, research for providing QoS-based services has sought for cross-layer collaboration in network stack [2]. But, these cross-layering efforts have only increased the complexity of the protocol stack, with the layers being much more dependent upon one other, and less modular than the original stack.

Application-specific network protocol design has been explored by some projects. Plexus [7] allows applications to achieve high performance with customized protocols, where the application-specific protocols are installed dynamically into the operating system kernel. But, Plexus needs the protocols to be written in Modula-2, a type safe language, and it runs only in the context of the SPIN extensible operating system [1]. Exokernel [6] presents an architecture to permit the application-specific customization of operating system abstractions, including the network resources. While Exokernel provides a suitable architecture for supporting application-specific protocol stack, it is unclear what set of network abstractions will be suitable and/or sufficient for the evolving WASN environment and WASN applications.

Berkeley motes use TinyOS [12] operating system. TinyOS uses active message based networking and provides only the minimal networking support to the applications because of the extreme hardware constraints of the motes. This limits the types of applications that can be efficiently supported. The network stack used by PicoNodes at Berkeley [26] identifies the need for many WASN specific situations (such as keeping the transport layer off the stack), but it still does not support application-specific adaptation and other WASN design goals.

VIII. CONCLUSION

This paper highlights the need for rethinking the traditional TCP/IP stack for the WASN environment. It shows that hop-to-hop requirements, with other distinguishing characteristics of WASN, demands a new stack layering. We present the design of a new protocol stack that is suitable for sensor applications. By making the protocol layers adaptable to the application needs dynamically, the new stack will allow exploiting the application specific requirements. By taking a specific application scenario, we qualitatively show how the new stack design can support the scenario better than the traditional stacks.

REFERENCES

- [1] B. N. Bershad, C. Chambers, S. J. Eggers, C. Maeda, D. McNamee, P. Pardyak, S. Savage, and E. G. Sirer. SPIN - an extensible microkernel for application-specific operating system services. In *ACM SIGOPS European Workshop*, pages 68–71, 1994.
- [2] A. Campbell, G. Coulson, F. Garcia, D. Hutchison, and H. Leopold. Integrated quality of service for multimedia communications. In *INFOCOM (2)*, pages 732–739, 1993.
- [3] A.-M. Chou and V. Li. Slot allocation strategies for TDMA protocols in multihop packet radio networks. In *Proceedings of INFOCOM 1992*, pages 710–716, 1992.
- [4] D. D. Clark and D. L. Tennenhouse. Architectural considerations for a new generation of protocols. In *Proceedings of the ACM symposium on Communications architectures & protocols*, pages 200–208. ACM Press, 1990.
- [5] J. Elson, L. Girod, and D. Estrin. Fine-Grained Network Time Synchronization using Reference Broadcasts. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation*, Boston, Massachusetts, December 2002.
- [6] D. R. Engler, M. F. Kaashoek, and J. O’Toole. Exokernel: An operating system architecture for application-level resource management. In *Symposium on Operating Systems Principles*, pages 251–266, 1995.
- [7] M. E. Fiuczynski and B. N. Bershad. An extensible protocol architecture for application-specific networking. In *USENIX Annual Technical Conference*, pages 55–64, 1996.
- [8] J. Gronkvist. Traffic controlled spatial reuse TDMA in multi-hop radio networks. In *Proceedings of 9th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 1203–1207, 1998.
- [9] T. He, J. A. Stankovic, C. Lu, and T. Abdelzaher. SPEED: A Stateless Protocol for Real-Time Communication. In *Proceedings of ICDCS 2003*.
- [10] J. Heidemann, F. Silva, and D. Estrin. Matching Data Dissemination Algorithms to Application Requirements. In *SenSys 2003*, Nov 2003.
- [11] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan. Building efficient wireless sensor networks with low-level naming. In *Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 146–159. ACM Press, 2001.
- [12] 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.
- [13] IEEE Computer Society LAN MAN Standards Committee. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std 802.11-1997. The Institute of Electrical and Electronics Engineers, New York, New York, 1997.
- [14] IEEE Computer Society LAN MAN Standards Committee. *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)*, IEEE Std 802.15.4. The Institute of Electrical and Electronics Engineers, New York, New York, 2003.
- [15] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings, Sixth Annual Int. Conf. on Mobile Computing and Networking (MobiCOM ’00)*, pages 56–67, Boston, Massachusetts, USA, 2000.
- [16] International Standard Organization. OSI - Basic Reference Model. ISO 7498, 1984.
- [17] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Mobile Computing and Networking*, pages 243–254, 2000.
- [18] C. A. Kent and J. C. Mogul. Fragmentation considered harmful. *WRL Technical Report 87/3*, 1987.
- [19] R. Kumar, M. Wolnetz, B. Agarwalla, J. S. Sin, P. W. Hutto, A. Paul, and K. Ramachandran. DFuse: Framework for Distributed Data Fusion. In *SenSys 2003*, Nov 2003.
- [20] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *Operating System Design and Implementation(OSDI)*, Boston,MA, Dec 2002.
- [21] R. Nelson and L. Kleinrock. Spatial-TDMA: A collision-free multihop channel access control. *IEEE Transactions on Computers*, 33:934–944, 1985.
- [22] S. PalChaudhuri, A. Saha, and D. B. Johnson. Adaptive clock synchronization in sensor networks. In *Proceeding of the Information Processing in Sensor Networks(IPSN)*, Berkeley, CA, April 2004.
- [23] K. Pentikousis. Tcp in wired-cum-wireless environments. *IEEE Communications Surveys & Tutorials*, vol. 3, no. 4, Fourth Quarter 2000.
- [24] R. Perlman. *Interconnections: Bridges, Routers, Switches, and Internetworking Protocols, 2/E*. Addison Wesley Professional, 2000.
- [25] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, 2000.
- [26] J. M. Rabaey and R. W. Brodersen. PicoRadio: Communication/Computation PicoNodes for Sensor Networks. DARPA Final Report, Dec 2002.
- [27] S. Singh and C. Raghavendra. PAMAS: Power Aware Multi-Access Protocol with Signalling for Ad Hoc Networks. *SIGCOMM Computer Communication Review*, 28(3), July 1998.
- [28] T. van Dam and K. Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the first international conference on Embedded networked sensor systems*, pages 171–180. ACM Press, 2003.
- [29] J. K. W. R. Heinzelman and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking*, pages 174–185, Seattle, WA USA, 1999.
- [30] W. Ye, J. Heidemann, and D. Estrin. An Energy-Efficient MAC protocol for Wireless Sensor Networks. In *Proceedings of INFOCOM 2002*, New York, New York, June 2002.
- [31] W. Yuen, H. Lee, and T. Andersen. A simple and effective cross layer networking system for mobile ad hoc networks. In *Proceedings of PIMRC*, 2002.