Random, Ephemeral Transaction Identifiers in Dynamic Sensor Networks*

Jeremy Elson and Deborah Estrin
Department of Computer Science, University of California, Los Angeles; and
USC/Information Sciences Institute
{jelson,estrin}@isi.edu

Abstract

Recent advances in miniaturization and low-cost, low-power design have led to active research in large-scale, highly distributed systems of small, wireless, low-power, unattended sensors and actuators. We explore the use of *Random, Ephemeral TRansaction Identifiers* (RETRI) in such systems, and contrast it with the typical design philosophy of using static identifiers in roles such as node addressing or efficient data naming. Instead of using statically assigned identifiers that are guaranteed to be unique, nodes randomly select probabilistically unique identifiers for each new transaction.

We show how this randomized scheme can significantly improve the system's energy efficiency in contexts where that efficiency is paramount, such as energy-constrained wireless sensor networks. Benefits are realized if the typical data size is small compared to the size of an identifier, and the number of transactions seen by an individual node is small compared to the number of nodes that exist in the entire system. Our scheme is designed to scale well: identifier sizes grow with a system's density, not its overall size. We quantify these benefits using an analytic model that predicts our scheme's efficiency. We also describe an implementation as applied to packet fragmentation and an experiment that validates our model.

1 Introduction

Recent advances in miniaturization and low-cost, low-power design have led to active research in large-scale, highly distributed systems of small, wireless, low-power, unattended sensors and actuators [1, 2, 3]. The vision of many researchers is to create sensor-rich "smart environments" through planned or ad-hoc deployment of thousands of sensors, each with a short-range wireless communications channel, and capable of detecting ambient conditions such as temperature, movement, sound, light, or the presence of certain objects.

While this new class of distributed systems has the potential to enable a wide range of applications, it also poses serious design challenges, described more fully by KKP [2] and EGHK [3]. The sheer number of these devices makes global broadcasting undesirable; wireless nodes' limited communication range relative to the geographic area spanned by the system often makes global broadcasting so inefficient that it is infeasible. As a consequence, many argue that nodes must depend on localized algorithms—making control decisions based solely on interactions with neighbors, without global system knowledge [3, 4].

Sensor networks must be *energy-efficient*. Most current network protocols are designed to be conservative only in their use of bandwidth or required processing. Many nodes in the emerging sensor systems will be untethered and therefore have small energy reserves. All communication—even passive listening—will have a significant effect on those reserves. Therefore, to maximize the lifetime of the system, it is critical to maximize the usefulness of *every bit* transmitted or received [5].

Sensor networks are expected to be *dynamic*. Over time, sensors may fail or new sensors may be added. Sensors will experience changes in their position, reachability, available energy, and even task details. These changes make configuration unacceptable; the system must evolve to make the best use of available resources.

Sensor networks must be *self-configuring*. Large-scale networks such as the Internet work in the face of brittle software partly because the number of people maintaining the network has grown along with the size of the

*Under submission to ICDCS 2001.

network itself. In contrast, there may be a single human responsible for thousands of nodes in a dense sensor network. It simply becomes impractical to imagine any design where each of these devices requires individual attention. In fact, manual configuration is ruled out completely if sensors are dropped into inhospitable terrain. This is expected in disaster-relief scenarios such as earthquakes and fires.

All of these distinguishing characteristics can potentially affect critical aspects of the system's design—routing and addressing mechanisms, naming and binding services, application architectures, security mechanisms, and so forth. Many of these problems are more challenging than their analogues in the traditional networking and distributed systems spaces, but there are also new opportunities for solutions.

This paper focuses on one such opportunity: a design philosophy for sensor networks where small, randomly selected, ephemeral, *probabilistically unique* transaction identifiers (which we call *RETRI*) are used in place of identifiers that are guaranteed to be unique. We define a "transaction" quite generally to be any computation during which some state must be maintained by the nodes involved in it. The identifier might be playing any of a number of roles; for example, a node's address, or the abbreviated form of a commonly-occurring bit sequence used in compression schemes. In traditional distributed systems, such identifiers are guaranteed to be unique and unambiguous. This guarantee comes with a number of important costs, described in the next section, that RETRI helps to avoid.

Most importantly, RETRI changes the scaling properties of a distributed system such that identifier sizes are tied to a system's *transaction density*, not its overall size. We define the transaction density as the average number of transactions that occur in the same place (connectivity-wise) and at the same time in a system. Many factors affect this value. Networks designed using spatially localized algorithms will have much lower transaction densities than those in which all nodes can potentially peer with each other. Higher physical node densities have an important effect because of the broadcast nature of the wireless medium. Temporal locality and the average duration of a transaction are also important: spatially local nodes that transmit simultaneously will see a higher transaction density than a group that spreads out its communication over time. Our scheme scales well because all of these factors can remain constant in a distributed system as it grows in size.

We show how RETRI can significantly improve the system's energy efficiency in contexts where that efficiency is paramount, such as energy-constrained wireless sensor networks. Benefits are realized if the typical data size is small compared to the size of an identifier, and the number of transactions seen by an individual node is small compared to the number of nodes that exist in the entire system.

Although our premise is applicable to many uses of identifiers, we studied it in the context of one particular use: unique node addresses. We explore the idea of ephemeral identifiers using an address-free packet fragmentation service as a case study. Our contributions are the description of this address-free method, a simple model that predicts its performance, and an implementation that validates our model. The next section will review the traditional role of network addresses and discuss their costs. Section 3 presents our address-free fragmentation. In Section 4, we analyze our architecture and develop a model to evaluate its performance. We describe an actual implementation and experiment used to validate our model in Section 5. Section 6 describes other contexts in which RETRI can be used. Related work is reviewed in Section 7. Finally, in Section 8, we present our conclusions.

2 The Use and Cost of Addresses

Inherent to the design of most distributed systems today is the assumption that each node has a unique network address. These addresses appear in every packet to identify its source and destination. We will consider special addressing requirements that a sensor network may have. However, it is useful to first reconsider the role that addresses play in traditional distributed systems.

2.1 The Role of Addresses in Traditional Distributed Systems

One of the primary purposes of an address in a traditional distributed system is to provide topological information that can be used to find routes. Addresses are sometimes also used as names in order to specify a communications endpoint: "I need to contact *that* node." These roles are the purest for an address according to the generally accepted definitions of names, addresses, and routes [6].

An important property of addresses in traditional distributed systems is that every node has a globally unique one. Addressing therefore has an additional benefit—essentially a side effect—of assigning *unique identifiers* to nodes. Unlike addresses, unique identifiers have no inherent meaning. They have no special properties other than their mutual distinctiveness. Many protocols are designed under the assumption that unique identifiers are available; addresses are sometimes used in this role simply because they are already there. We will illustrate this point with two examples:

- IP [7] specifies a way of fragmenting a datagram into smaller packets for transport across physical networks with small frame sizes. During fragmentation, IP sources give datagrams locally unique identification numbers.[1] This allows the 4-tuple of (Source IP Address, Destination IP Address, Identification Number, Protocol Number) to be used as a *unique datagram identifier.* IP fragmentation and reassembly therefore ignore the significance of a source or destination address as an *address.*

- In TCP [8], a *flow* is uniquely identified by the 4-tuple of (Source Address, Source Port, Destination Address, Destination Port). This tuple is used to demultiplex incoming packets back into their constituent flows. The destination address is, of course, used to route packets. However, the demultiplexing stage treats the tuple only as a unique *flow identifier.* As in the fragmentation example, the structure and inherent meaning of the address are ignored.

In a system where each node has a unique address, allowing those addresses to serve the dual role of unique identifiers is often a good idea and a natural choice. As we will discuss in Section 3, a key to our architecture is separating these two roles. Others have objected to the overloaded role of addresses for various reasons; Chiappa's "Nimrod" architecture [9] is one example. We chose to separate the roles because unique addresses come with a price. In fact, in some contexts, they can be quite expensive.

2.2 Global vs. Local Addresses

Most addressing schemes can be classified as using either *global* or *local* addresses. Each has its own advantages and disadvantages.

In systems such as Ethernet [10], every node that exists has a globally unique address. The address is statically assigned, typically at the time of the device's manufacture. This approach guarantees that any particular collection of interconnected Ethernet devices will have distinct addresses. However, it can be inefficient if the number of devices that exist is much larger than the number that are interconnected. Ethernet's large data payloads and lack of energy constraints make this inefficiency small, and certainly well worth the convenience that it buys. In contrast, we will argue in the next section that the cost of globally unique addresses is much higher in a sensor network.

System	Address Bits
IPv4	32
Ethernet	48
IPv6	128

An alternative is to configure nodes with addresses that are locally unique. That is, each node in a distributed system has an address that is unique with respect to its potential peers, based on the connectivity of the network or the scope of communication. Devices that are mutually disconnected may share the same address at the same time. This is in contrast to globally unique addresses that are always distinct with respect to every other device that exists.

Global address spaces tend to be very large to make decentralized allocation more convenient, and to accommodate growth in the number of nodes. If the number of interconnected devices is significantly smaller than the total number of devices, locally unique addresses can use far fewer bits than globally unique addresses. For example, although Ethernet nodes use 48 bit addresses, a typical Ethernet of a few hundred nodes could use only 10 or fewer bits if the addresses were locally assigned. In contrast, such savings are not possible in IP addressing because every node on the Internet can potentially communicate with every other node.[2]

Different methods exist for assigning local addresses. The simplest example is manual configuration. Protocols such as DHCP [11] allocate addresses from a local authority. More complex schemes such as the multicast address allocation in SDR [12] and MASC [13] listen to the addresses already in use, allocate addresses autonomously, then detect collisions and resolve conflicting address claims. They also use explicit scoping to achieve spatial reuse of addresses.

[1]This is not the same as TCP's sequence number.

[2]This analogy ignores other properties of IP addresses such as their topological significance. We also ignore private IP networks, firewalls, and the like.

2.3 The Cost of Addresses in Sensor Networks

Complementary to the discussion of the benefit of an address must be a discussion of its cost. By the *cost* of an address, we refer to the overhead required in terms of network utilization. It is vital to consider this cost in unattended wireless sensor networks where, as Pottie points out, *every bit transmitted reduces the lifetime of the network* [5]. We do not attempt to quantify factors such as the human effort required to coordinate a global address allocation.

A key factor to consider is that both the packet size and data rate in many typical sensor networks will be very small. This is by design: the energy cost of communication makes it desirable for nodes to minimize the size and frequency of transmissions by doing as much local processing, summarization, and aggregation of data as possible. Although sensors will transmit large messages occasionally, we expect that they will normally transmit periodic messages consisting of only a few bits to describe the current state—perhaps the ambient temperature or the number of vehicles detected in the past hour.

The cost of an address in an energy-constrained distributed system can be considered high if the address space is underutilized and the address itself accounts for a significant portion of the total number of bits transmitted. In sensor networks, globally unique addresses would need to be very large—at least as large as Ethernet's 48 bits—compared to the typical few bits of data attached to them. Therefore, local addressing seems to be needed.

To maintain local addresses, a sensor network could use a protocol that dynamically assigns addresses to nodes based on the addresses of other nodes in the neighborhood. However, as the network topology becomes more dynamic, more work is required to keep addresses locally unique. This scheme will be efficient only as long as the address-allocation overhead is small compared to the amount of useful data transmitted. In a static system, the work done at the beginning to resolve address conflicts is amortized over all the work done in the system thereafter. In sensor networks, the expected dynamics make this scheme potentially very inefficient given the low data rate.

The other methods for assigning locally unique addresses we discussed in Section 2.2 are also inappropriate for sensor networks. Manual configuration is clearly not possible for reasons we described in Section 1. A central address authority is not possible because of the highly decentralized nature of the network and the lack of global knowledge at any single node.

3 Address-Free Fragmentation

The previous section seems to paint a bleak picture of our options for designing an efficient addressing scheme for sensor networks. Large packets or high data rates provide plenty of bits over which to amortize either the cost of addresses that are long enough to be globally unique, or the cost of running a protocol that assigns locally unique addresses. We are not so lucky in low data rate distributed systems with high dynamics and energy constraints.

The SCADDS project [3] provides a potential framework for a solution because is uses *attribute-based data naming* [14, 15, 16]. In SCADDS, applications are unlikely to ask the question: "Was there motion detected at sensor #27.201.3.97?" Rather, they might ask: "Was there motion detected in the north-east quadrant?" or "Where has motion been detected recently?" This kind of data naming will be application-specific, and effectively moves naming, addressing, and even routing from the network layer into the *application*.

We wondered, if unique addresses are only being used as unique *identifiers*—and their significance as *addresses* are being ignored—why are they still sent in each packet?

3.1 Randomized Transaction Identifiers

Our idea, at its core, is very simple: whenever a guaranteed–unique identifier is needed, an ephemeral, randomly selected, *probabilistically–unique* identifier can be used instead. Of course, there is a chance that two peers will use the same identifier at the same time. We do not try to resolve such conflicts. Identifier collisions lead to lost transactions, and are treated like any other loss. Sensor networks already must be highly robust to existing common sources of loss such as RF collisions and node or environment dynamics that change connectivity. Occasional losses due to identifier collisions are likely to have a negligible marginal effect considering the losses that sensor networks must already assume. By choosing a new random identifier for *each transaction*, persistent losses are avoided.

This scheme is best illustrated with an example. In Section 2.1, we described IP packet fragmentation and noted that it depends on having a unique packet identifier. IP uses the sender's unique IP address plus an identifier generated by the sender. In our Address-Free Fragmentation ("AFF"), each packet simply receives a random packet identifier. Once an identifier is selected for a packet, all of that packet's fragments receive the same identifier, allowing receivers to correctly reconstruct the packet. The next packet receives a new random identifier. For this service, we define a "transaction" as the transmission of all of a single packet's fragments.

Often, an identifier is simply a way to reference state that is kept on a transmitter or receiver. The identifier provides continuity among the packets that make up a logical transaction. Fragmentation is an example that illustrates the identifier's role. By separating the function of an address from that of a unique identifier, we free nodes from using addresses in situations when a unique identifier is really all that is needed.

Because new identifiers are selected for each packet, the AFF identifier alone is not sufficient to tell a receiver which node sent a packet, or even if two successive packets were sent by the same node. Of course, there may still be situations when a specific node needs to be identified—for example, for debugging or maintenance purposes. Long, globally unique identifiers, statically assigned like Ethernet hardware addresses, are appropriate for this. In AFF, we are not arguing against *assigning* globally unique identifiers to nodes. Rather, we are suggesting that unique identifiers should be used sparingly—even if they have been assigned—in favor of AFF identifiers that are likely to be much shorter. A node's unique identifier can be sent as *data*, on demand, instead of in the header of every packet.

3.2 Scoping and Listening

For scalability, interactions in sensor networks are being designed to be localized. The energy cost of communication makes local processing and summarization at each node far more efficient than simply forwarding large amounts of data end-to-end through many hops [5]. AFF takes advantage of this spatial locality of sensor networks: nodes that are far apart may use the same identifier at the same time. AFF also takes advantage of temporal locality: nearby nodes can use the same identifier at different times. AFF identifiers must only be unique to a particular place at a particular time. In contrast, globally unique addresses must be unique with respect to every other node that exists. By leveraging locality, the size of AFF identifiers must only scale with the transaction density of a growing sensor network, while a global address space must scale with the total number of nodes in the network. For these reasons, we expect that AFF identifiers can be much shorter than globally unique addresses.

One heuristic that can significantly improve the performance of the scheme is *listening*. That is, instead of picking identifiers completely at random, nodes can avoid using identifiers that are already in use by listening to packets being transmitted. This is not guaranteed to work perfectly, of course: two nodes that are not in range of each other might pick the same identifier when trying to communicate with a receiver that lies in between them.[3] (To help alleviate this problem, the receiver could try to send an explicit "identifier collision notification" to the two senders.)

Packet loss may also prevent perfect listening. In addition, some nodes may choose to minimize the time they spend listening because of the significant power requirements of running a radio. Because of these limitations, listening is usually not has helpful as making the size of the identifier pool larger, but it does help to make the best possible use of available resources. Listening has been successfully used to reduce the probability of address allocation collisions in other contexts such as SDR [12], a tool for distributed allocation of Internet multicast addresses.

4 Analysis

In this section we describe a simple analytic model of both AFF and static address allocation. The goal is to develop a model that will predict the efficiency of AFF compared to a static address allocation. Our model is not sufficiently general to predict the performance in all possible scenarios, but it does provide a basis for comparison between the two architectures.

[3]The wireless literature often refers to this as the *hidden terminal problem.*

4.1 A Model of AFF

The first step in our analysis is to define a simple efficiency metric. Our metric is essentially the "cost-benefit ratio" of transmitting bits:

$$E = \frac{\text{Useful Bits Received}}{\text{Total Bits Transmitted}} \tag{1}$$

In our model, bits are transmitted in *packets*, each of which is made up of a *header* and *data*. The cost of a packet is the cost of transmitting both its header and data. The header is not considered inherently *useful*, but rather something that facilitates the transmission of potentially useful data.

Every packet is part of a *transaction*. We assume that the transaction density, T, is the average number of concurrent transactions visible at any single point in the network. A limitation of our model is that the single parameter T is not sufficient to describe all possible scenarios—two long transactions will have different collision characteristics than a long transaction competing with a series of short transactions, even though $T = 2$ in both cases. In order to simplify our analysis, at the expense of making our model less general, we will assume that every transaction spans the same amount of time.

In our model, a packet header consists of solely a transaction identifier. At the core of our model is the assumption that a transaction is successful (that is, useful) if and only if the source uses an identifier that is unique with respect to all other transactions at the same point in the network for the entire duration of the transaction. Transactions that fail due to identifier collisions reduce efficiency because the cost of transmitted bits is incurred without the benefit of the reception of useful bits.

In the case of static allocation, where every node is given a distinct address, we assume that identifier collisions (and, therefore, failed transactions) are impossible. If, on average, we transmit D bits of data with an H-bit header (address), the efficiency is simply:

$$E_{static} = \frac{D}{D + H} \tag{2}$$

Equation 2 expresses the ratio of data bits to total bits for an entire transaction, not just a single packet.

In our address-free architecture, successful transactions are no longer guaranteed. Transactions only have some *probability* of success due to possible identifier collisions. We assume that the entire transaction will either succeed or fail, causing either the reception or loss of all its constituent packets. Keeping in mind our assumption that all transactions are the same length, we arrive at a different efficiency metric:

$$E_{afa} = \frac{D \times P(\text{Success})}{D + H} \tag{3}$$

The probability of a successful transaction given in Equation 3 is dependent on three factors. The first is the the number of concurrent transactions T that are contending for identifiers. Based on our previous assumption that all transactions are the same length, each transaction will overlap with the beginning or end of $2(T-1)$ other transactions in the worst case. The second factor is the size of the space from which the identifiers are drawn, 2^H for an H-bit identifier. The third factor is the algorithm used for selecting the identifiers. Heuristics for reducing the probability of identifier collisions were described in Section 3. However, we will analyze the simplest and most pessimistic scenario in which every node picks its transaction identifiers uniformly from the identifier space without regard to any learned state. This makes the analysis straightforward because each node's identifier selections are independent:

$$P(\text{Success}) = \left(1 - \frac{1}{2^H}\right)^{2(T-1)} \tag{4}$$

Equation 4 is useful in that it gives a reasonable upper bound on the expected probability of identifier collisions. Heuristics such as listening can improve significantly on this bound in practice, as we will show in Section 5. It can be seen in Equation 3 that a significant increase in the probability of successful transactions leads directly to a significant increase in the efficiency of AFF.

4.2 Comparison of AFF to Static Allocation

We now wish to use our model to compare AFF over a range of identifier sizes with static allocation. In doing so we must first decide on a reasonable number of bits that should be used in static allocation for the purposes of comparison.

We assume that sensor networks will consist of tens of thousands of nodes. If addresses are assigned optimally, about 16 bits will be sufficient to address the entire network. On the other hand, past experience has shown that optimal address allocation is often difficult to achieve, or undesirable because it makes decentralized allocation difficult. The 48-bit address used in Ethernet is designed for distributed assignment of a single, universal address space. It is also meant to be sufficient for the total number of devices that *exist*, even though the number of nodes that make up any specific Ethernet network is much smaller. Hardware in sensor networks may be given similar global addresses—perhaps with an even larger address space given the expected scale and density of their deployment. However, to be conservative in our analysis, we will also compare AFF to 32-bit static addresses.

Figure 1 compares static allocation of 16- and 32-bit identifiers to AFF over a range of identifier sizes. In this example, the size of the data is 16 bits. Three different scenarios are considered for AFF: cases where 16, 256, and 65,536 transactions are simultaneously visible to individual nodes in the network. The total number of concurrent transactions in the entire network may be much larger and is not part of the model.

In the static case, transmitting 16 bits of data with a 16- or 32-bit identifier always leads to a constant 50% or 33% efficiency, respectively. These cases are represented by the flat lines in the figure. The curves describing AFF also have a consistent shape. If the number of identifier bits is too small, efficiency is low due to a large number of identifier collisions. As the number of identifier bits increases, the probability of identifier collisions quickly drops close to zero and the efficiency is dominated by the ratio of data bits to total bits (as in Equation 2). The peak of the curve represents the optimal balance between two opposing goals: minimizing the number of collisions and minimizing the number of header bits transmitted per data bit.

In a sensor network with tens of thousands of nodes, there might be only tens or hundreds of simultaneous transactions visible at any one place at any one time. Our model quantifies the intuition that AFF is more efficient if this locality of transactions allows AFF to user fewer bits than static allocation. In other words, AFF can be more efficient if locality causes a global address space to be underutilized. For example, as shown in Figure 1, AFF works optimally with only 9 identifier bits in a network where there are an average of 16 simultaneous transactions seen by any node. This is more efficient than a static assignment that might need 16 or 32 bits. On the other hand, in an extreme (and, we believe, very unlikely) case of 64K simultaneous transactions seen by every node in a 64K node network, there is no room for AFF to improve; a 16-bit address space can be fully (indeed, optimally) utilized.

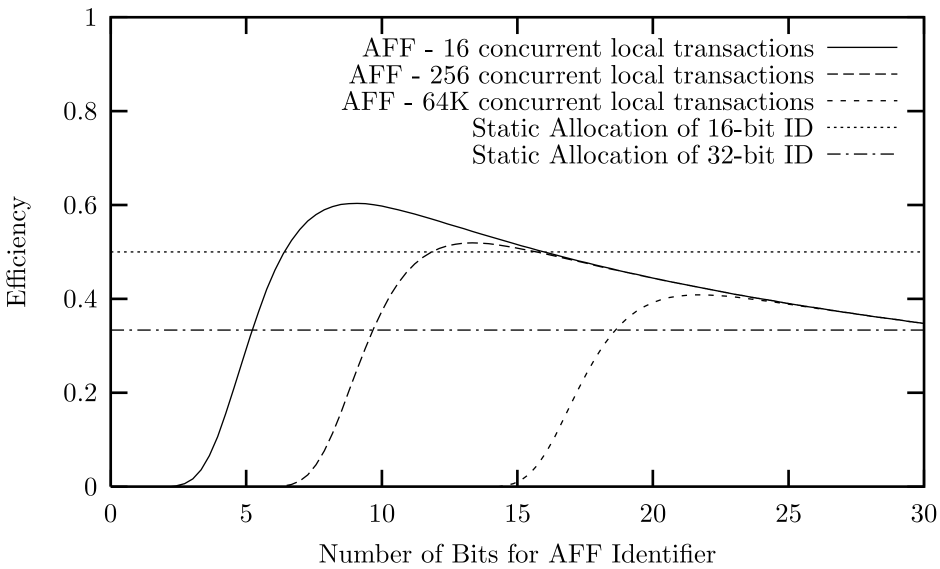

Figure 1: Efficiency of AFF vs. static allocation for 16-bit data.

Figure 2 shows the same analysis assuming 128 bit data rather than 16. There are a number of interesting features to note. First, the larger data size makes static allocation more efficient because the same number of address bits are now being amortized over a larger number of data bits. Second, the optimal number of bits used for the AFF identifier increases. Intuitively, this is because the larger data size makes the cost of a collision higher; the increased cost of transmitting more identifier bits is worth the benefit of a lower collision rate.

4.3 Advantages of AFF

AFF is fundamentally different from static address allocation because the size of AFF's identifier space is tied to the transaction density of the network, as opposed to static allocation whose space is commensurate with the total *size* of the network. A node using AFF needs to select an identifier that is unique with respect to its local neighbors; the potentially large network that lies beyond a node's limited point of view becomes immaterial. AFF can take advantage of spatial and temporal locality to use fewer identifier bits than would be required for a static, globally unique allocation.

Figure 1 shows one benefit of this difference. Based on our assumptions of reasonable numbers of network densities and sizes, AFF works optimally with only 9 bits while static allocation needs 16 or 32. When amortized over only 16 bits of data, AFF can result in a increase in efficiency and thus network lifetime.

In Figure 2, which assumes larger data, the differences are less pronounced. At this design point, the efficiency of AFF and static allocation are not significantly different. However, using AFF still buys us vastly better scaling properties. As the network grows, AFF's optimum identifier size will remain the same. In contrast, the size of identifiers for globally unique static allocation must grow with the size of the network.

Clearly, AFF does not help in networks that do not exhibit locality. If the optimum number of bits needed by AFF is the same as the number of bits required for globally unique static allocation, traditional static allocation will always be better. This point is illustrated in Figure 3. The figure shows our model from a different perspective, describing how efficiency of the network changes as the load on it increases. Statically assigned identifiers have constant efficiency until the address space is exhausted, after which the efficiency is undefined. AFF does work beyond this point, though networks should never be so severely underprovisioned by design.

4.4 Does Fewer Bits Mean Less Energy?

We have mentioned that energy savings are one of the primary motivations for using shorter identifiers. However, the actual energy savings achieved by reducing the number of bits transmitted is highly dependent on details of the radio hardware and MAC protocol in use. While 20 bits saved by AFF may have a large impact on efficiency

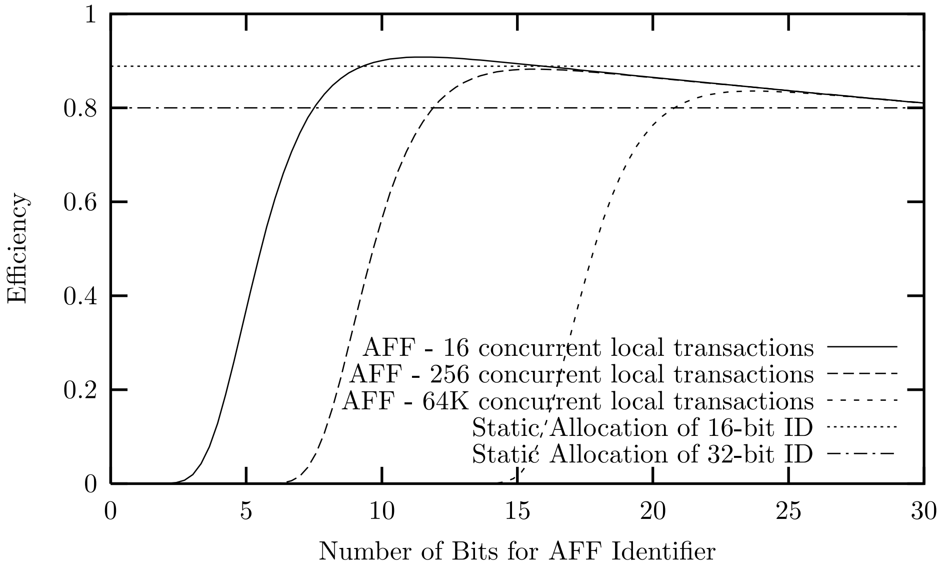

Figure 2: Efficiency of AFF vs. static allocation for 128-bit data.

in the user-data portion of a packet (as seen in Figure 1's example), that savings becomes meaningless if used with a MAC layer such as 802.11 that adds hundreds of bits of overhead per packet. In the higher-powered regime of laptops running complex MAC protocols, AFF makes little sense.

However, these few bits do become much more meaningful in the context of the very low-powered radios designed for sensor networks—such as those made by Radiometrix, RF Monolithics, and perhaps even the forthcoming Bluetooth radios. These radios have extremely simple MACs and framing that leads to a more direct correlation between the amount of user data sent to the radio and the energy expended to send it. Quantifying this relationship is a subject of our ongoing research.

These lower-powered radios are also often associated with limited frame sizes. The Radiometrix RPC radio, for example, can transmit only 27 bytes per frame. AFF originally came from a desire to squeeze as much information as possible into these small frames.

5 Implementation

We have implemented an AFF-based packet fragmentation service for use with low-power, short-range wireless radios. Our implementation was born out of a necessity to send large virtual packets over radios with a very small (27 byte) frame size, but it also served as an important proof of concept for AFF and a validation of the analytic model presented in Section 4.

Our current testbed consists of Radiometrix RPC 418 MHz radio modules[4] attached to Toshiba Libretto laptops running the RedHat Linux operating system. The RPC is our radio of choice because of its low power requirements, ease of use, and small form factor. A simple packet controller on the radio accepts frames of up to 27 bytes from the host and attempts to broadcast each frame to all other RPC modules within receiving range. Any RPC in the area that successfully receives the frame transfers it back up to its host.

Our fragmentation driver accepts packets of up to 64Kbytes from applications, fragments them to fit into 27 byte frames, and sends them down to the RPC for transmission. It also watches for fragments coming in from the radio, reassembles them, and delivers successfully reconstructed packets to the host.

The fragmentation algorithm itself is simple, based on IP fragmentation. Each packet is given a random AFF identifier. A "packet introduction" fragment is transmitted first, containing the packet's AFF identifier, total length, and checksum. Each fragment is then transmitted with the packet's AFF identifier and the byte offset of the data it carries. The driver on the receiver reassembles fragments as they are received and delivers packets

[4] More information available at `http://www.radiometrix.co.uk`

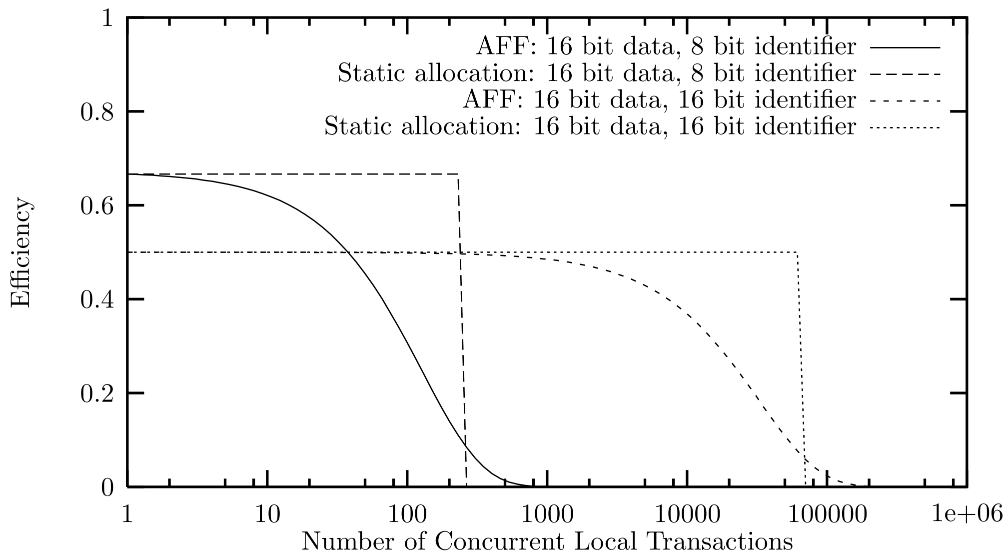

Figure 3: Efficiency of AFF vs. static allocation for 16-bit data.

to the application when the checksum succeeds. Packets that suffer from identifier collisions are never delivered because of checksum failures or other inconsistencies.

5.1 Experiments

We created a specially instrumented version of our fragmentation driver in an attempt to validate our model of predicted collision rates given in Equation 4. In the instrumented driver, each node has a globally unique identifier; the fragment format is augmented to include this identifier along with the randomly selected AFF identifier. By examining both the AFF identifier and the guaranteed unique node identifier of received fragments, the receiver's driver is able to determine how many packets would have been lost due to AFF identifier collisions if the unique ID had not been present.

Our experimental testbed consisted of five radio transmitters simultaneously sending packets to a single wireless receiver. We tested this configuration over a range of AFF identifier sizes. Ten trials were executed for each identifier size. In each trial, each of the five transmitters attempted to transmit a continuous stream of random 80-byte packets for two minutes; each of these packets were fragmented into five fragments (a single fragment introduction and four data fragments). After attempting to reassemble all received fragments, the receiver reported the total number of packets successfully received using the packet's unique identifiers and the number that would have been received based on the AFF identifier alone.

We also tested the efficacy of a simple listening heuristic to reduce identifier collisions. In the listening mode, each transmitter also acts as a receiver, listening to packets transmitted by other nodes. When selecting an AFF identifier for outgoing packets, transmitters did not use identifiers they had recently heard in use by other transmitters. The choice of identifier was picked uniformly from pool of not-recently-used identifiers. We adaptively define "recently" as within the most recent $2T$ transactions; each node can estimate T based on the number of concurrent transactions it observes. All of the transmitters and receivers were arranged so that they were fully connected (i.e., all the radios were well in range of each other).

The results of these experiments is shown in Figure 4 along with our model's prediction (for $T = 5$). As evident in the figure, our collision model appears to be accurate. The simple listening algorithm also appeared to be very effective in reducing identifier collisions.

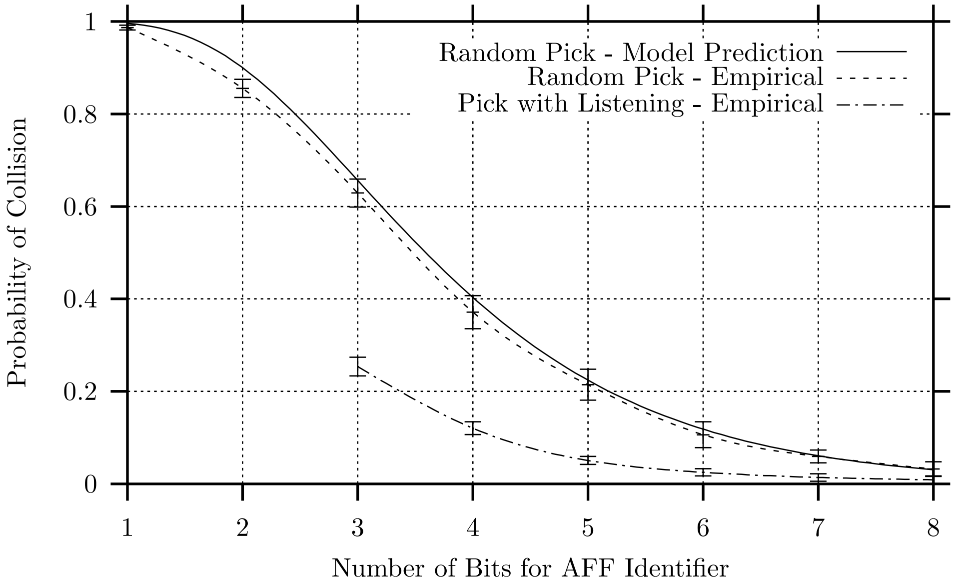

Figure 4: Collision rate predicted by our model vs. actual collision rates observed in our implementation. We tested one algorithm that selects identifiers randomly and one that attempts to avoid collisions by listening to other nodes' transmissions. The error bars represent the standard deviation from the mean for each trial.

6 RETRI in Other Contexts

The idea of using RETRI, exemplified in our address-free fragmentation, has applications in other areas of a distributed sensor network. These applications all have in common a need to reference some state that has meaning over some time period and in some location. Each application's transaction density is affected by the temporal and spatial extent over which this state must remain valid. The definition of a transaction and the method of detection of collisions are both highly application-dependent.

A number of other examples are seen in sensor networks, described in more detail in [3], including:

- *Interest reinforcement.* Nodes that periodically transmit their sensor readings may wish to transmit more or less frequently depending on the network's level of interest in their readings. When a node transmits a sensor reading, its neighbors periodically send feedback to the transmitter indicating their level of interest. With unique addresses assigned to each transmitter, the feedback might take the form of a message such as "Sensor #27.201.3.97, send more of your data." An address is not actually needed in this context; it is simply used as a way of describing data that was previously received. RETRI can serve this purpose equally well: "Whoever just sent data with Identifier 4, send more of that."
- *Attribute-based name compression.* The attributes and associated values might be quite large, but the same attribute/value pairs might be used frequently by a node. This problem has traditionally been solved by creation of a "codebook" mapping small identifiers to long lists of attributes. Nodes using codebooks can choose RETRI identifiers instead of traditional alternatives (using large, guaranteed-unique identifiers, or expending energy to ensure that the codes are conflict-free).

Although identifier conflicts can lead to losses or unexpected behavior, robustness to these types of errors *must already be fundamental to the design of these systems*, where errors are the norm due to factors such as node dynamics, changes in the environment, and the vagaries of RF connectivity. Occasional identifier collisions will have a small marginal effect, and persistent or systematic collisions are avoided by picking a new identifier for each new transaction.

7 Related Work

Our work may be most similar to the use of session directories for Internet multicast address allocation described by Handley [12]. Multicast addresses are selected randomly in a decentralized manner by nodes that need to establish multicast groups. The idea parallels AFF's use of temporal locality (multicast addresses are reused over time) and spatial locality (TTL scoping prevents multicast groups from being globally visible). In contrast, our focus on sensor networks brings with it different constraints such as the paramount importance of energy efficiency.

In the Nimrod architecture [9], Chiappa recognizes problems that arise when addresses serve multiple, overloaded roles. He suggests an architecture in which nodes have a globally unique endpoint identifier separate from their unique and topologically significant "locator." This allows endpoints to have a stable identity regardless of changes in the network topology. In contrast, our work is more concerned with the cost of identifiers long enough to be globally unique in context where locally unique identifiers are sufficient. Our scheme also assumes some other method is used for naming, and names need not be unique.

In WINS [1], Kaiser and Pottie have designed a system where short, locally unique addresses are dynamically assigned to nodes in a radio cluster by a central controller. They try to maximize the use of energy in a wireless, unattended radio system by reducing the number of address bits transmitted. Their motivation is similar to AFF; however, AFF's design does not require centralized cluster formation. This makes AFF more scalable, feasible without a centralized controller, and robust in the face of high dynamics.

In the MIT Amorphous Computing project, Coore *et. al.* have described algorithms for hierarchy construction that rely on randomized node identifiers to break ties in elections [17]. Our work, in contrast, is oriented towards identifiers used to facilitate transactions, and does not deal with cluster formation.

In SCADDS [3], EGHK propose an architecture for scalable coordination and control in deeply distributed systems such as sensor networks. Our work is complementary to SCADDS and we owe many of our assumptions about future sensor network architectures to that project. The attribute-based data naming proposed in SCADDS is similar to the naming schemes used by Raman and McCanne in ALF [14, 18], Adjie-Winoto *et. al.* in the Intentional Naming System [15], and Michel *et. al.* in their adaptive web caching architecture [16].

8 Conclusions and Future Work

We have presented a rationale for using *Random, Ephemeral TRansaction Identifiers* (RETRI) in sensor networks. In our scheme, nodes pick semi-random identifiers that are used for the duration of a single transaction. Occasionally, identifier conflicts lead to losses, which have a small marginal effect on a network which must already be highly robust to losses due to the vagaries of RF connectivity or node dynamics. Persistent identifier collisions are avoided by picking a new identifier for each new transaction. For many applications, we believe this technique is superior to the alternatives: statically-allocated, globally unique node identifiers are typically much longer, and protocols that dynamically assign locally unique addresses may be highly inefficient given the high rate of dynamics in sensor networks and low data rate over which to amortize the cost of the protocol.

We have also developed a simple model for predicting the rate of identifier collisions in a distributed system that uses RETRI. By predicting collisions we are able to model the overall efficiency of RETRI versus a static allocation policy for a given identifier size, data size, and average number of concurrent transactions. These models have been validated through experiments using an actual implementation.

Our models suggest that RETRI is superior to alternatives in distributed systems with the following properties:

- The system exhibits significant spatial or temporal locality. Specifically, the number of nodes that exist is far greater than the number of simultaneously communicating peers any individual is likely to see.
- The overall data rate is low, or the size of a typical packet is small compared to the size of a static address.
- Efficiency is paramount—for example, due to energy constraints.

RETRI improves the scaling properties of such distributed systems by allowing the size of the identifier space to grow as a function of the system's transaction density, rather than its overall size. RETRI can also significantly reduce the overhead required to transmit data by optimizing the number of header bits transmitted per data bit. By the same token, RETRI does not help in distributed systems that do not exhibit locality; if the optimum number of bits needed by RETRI is the same as the number of bits required for globally unique static allocation, traditional static allocation will always be better.

Although our model was successful in predicting performance in our simple testbed, predicting performance in a more complex system will be more difficult. In our ongoing research, we are refining our analysis of RETRI's expected performance and continuing larger-scale experiments. Specifically, we are interested in capturing the effects of listening and non-uniform transaction lengths in our model. A model of the system topology will be required to capture the effect of listening so that problems such as hidden terminal effects are taken into account. We are also investing more accurate ways of estimating the typical transaction density T.

Because the utility of RETRI increases as the transaction density decreases, RETRI is most useful in distributed systems that exhibit the highest degrees of locality in their interactions. We are therefore also investigating techniques for maximizing locality in sensor network interactions.

Acknowledgments

The authors are grateful to Ashish Goel for help in the development of our analysis. Lewis Girod also made contributions to our analysis in addition to helping to build our experimental testbed.

This work was supported by DARPA under grant No. DABT63-99-1-0011 as part of the SCADDS project, and was also made possible in part due to support from Cisco Systems.

References

[1] G. Asada, M. Dong, T.S. Lin, F. Newberg, G. Pottie, W.J. Kaiser, and H.O. Marcy. Wireless Integrated Network Sensors: Low Power Systems on a Chip. In *Proceedings of the European Solid State Circuits Conference*, 1998.

[2] J.M. Kahn, R.H. Katz, and K.S.J. Pister. Next century challenges: mobile networking for Smart Dust. In *Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking*, pages 271–278, 1999.

[3] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking*, pages 263–270, 1999.

[4] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking*, pages 56–67, Boston, MA, August 2000. ACM Press.

[5] G.J. Pottie and W.J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, May 2000.

[6] J.H. Saltzer. Naming and Binding of Objects. In R. Bayer, editor, *Operating Systems—An Advanced Course*. Springer–Verlag, 1978.

[7] J. Postel. RFC 791: Internet Protocol, September 1981.

[8] J. Postel. RFC 793: Transmission Control Protocol, September 1981.

[9] N. Chiappa. RFC 1753: IPng technical requirements of the Nimrod routing and addressing architecture, December 1994.

[10] R. M. Metcalfe and D. R. Boggs. Ethernet: Distributed packet switching for local computer networks. *Communications of the ACM*, 26(1):90–95, January 1983.

[11] R. Droms. RFC 1541: Dynamic host configuration protocol, October 1993.

[12] Mark Handley. Session directories and scalable Internet multicast address allocation. In *Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 105–116, 1998.

[13] Satish Kumar, Pavlin Radoslavov, David Thaler, Cengiz Alaettinoğlu, Deborah Estrin, and Mark Handley. The MASC/BGMP architecture for inter-domain multicast routing. In *Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 93–104, 1998.

[14] Suchitra Raman and Steven McCanne. Scalable data naming for application level framing in reliable multicast. In *Proceedings of the 6th ACM international conference on Multimedia*, pages 391–400, 1998.

[15] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley. The design and implementation of an intentional naming system. In *Proceedings of the 17th ACM Symposium on Operating System Principals*, December 1999.

[16] Scott Michel, Khoi Nguyen, Adam Rosenstein, Lixia Zhang, Sally Floyd, and Van Jacobson. Adaptive Web caching: towards a new global caching architecture. *Computer Networks And ISDN Systems*, 30(22-23):2169–2177, November 1998.

[17] D. Coore, R. Nagpal, and R. Weiss. Paradigms for Structure in an Amorphous Computer. A.I. Memo 1614, Massachusetts Institute of Technology, 1997.

[18] Sally Floyd, Van Jacobson, Steve McCanne, Ching-Gung Liu, and Lixia Zhang. A reliable multicast framework for light-weight sessions and application level framing. In *Proceedings of the ACM SIGCOMM '95 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 342–356, 1995.