Time Synchronization for Wireless 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

Time synchronization is a critical piece of infrastructure for any distributed system. Distributed, wireless sensor networks make extensive use of synchronized time, but often have unique requirements in the scope, lifetime, and precision of the synchronization achieved, as well as the time and energy required to achieve it. Existing time synchronization methods need to be extended to meet these new needs. We outline the synchronization requirements of future sensor networks and present an implementation of our own low-power synchronization scheme, post-facto synchronization. *We also describe an experiment that characterizes its performance for creating short-lived and localized but high-precision synchronization using very little energy.*

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, 7, 4]. 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.

Time synchronization is a critical piece of infrastructure for any distributed system. Distributed, wireless sensor networks make particularly extensive use of synchronized time: for example, to integrate a time-series of proximity detections into a velocity estimate [3]; to measure the time-of-flight of sound for localizing its source [5]; to distribute a beamforming array [13]; or to suppress redundant messages by recognizing that they describe duplicate detections of the same event by different sensors [6]. Sensor networks also have many of the same requirements as traditional distributed systems: accurate timestamps are often needed in cryptographic schemes, to coordinate events scheduled in the future, for ordering logged events during system debugging, and so forth.

The broad nature of sensor network applications leads to timing requirements whose scope, lifetime, and precision differ from traditional systems. In addition, 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. Time synchronization methods for sensor networks must therefore also be mindful of the time and energy that they consume.

In this paper, we argue that the heterogeneity of requirements across sensor network applications, the need for energy-efficiency and other constraints not found in conventional distributed systems, and even the variety of hardware on which sensor networks will be deployed, make current synchronization schemes inadequate to the task. In sensor networks, existing schemes will need to be extended and combined in new ways in order to provide service that meets the needs of applications with the minimum possible energy expenditures.

In this framework, we present our idea for *post-facto synchronization*, an extremely low-power method of synchronizing clocks in a local area when accurate timestamps are needed for specific events. We also present an experiment that suggests this multi-modal scheme is capable of precision on the order of 1μsec—an order of magnitude better than either of the two modes of which it is composed. These results are encouraging, although still preliminary and performed under idealized laboratory conditions.

In Section 2, we present a number of metrics that can be used to classify both the types of service provided by synchronization methods and the requirements of applications that use those methods. Section 3 describes our post-facto synchronization idea and presents an experiment that characterizes its performance. Future work is described in Section 4, and our conclusions are drawn in Section 5.

*To appear in Proceedings of the 2001 International Parallel and Distributed Processing Symposium (IPDPS), Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing, April 2001, San Francisco, CA, USA. Also published as UCLA CS Technical Report 200028.

2. Characterizing Time Synchronization

Many different methods of distributed time synchronization are in common use today. Systems such as the U.S. Global Positioning System (GPS) [8] and the WWV/WWVB radio stations operated by the National Institute of Standards and Technology [2] provide references to the U.S. time and frequency standards. Network time protocols, most notably Mills' NTP [10], distribute time received from these primary sources to network-connected computers.

In studying their application to sensor networks, we have found it useful to characterize the different types of time synchronization along various axes. We consider certain metrics to be especially important:

- *Precision*—either the dispersion among a group of peers, or maximum error with respect to an external standard.
- *Lifetime*—which can range from persistent synchronization that lasts as long as the network operates, to nearly instantaneous (useful, for example, if nodes want to compare the detection time of a single event).
- *Scope and Availability*—the geographic span of nodes that are synchronized, and completeness of coverage within that region.
- *Efficiency*—the time and energy expenditures needed to achieve synchronization.
- *Cost and Form Factor*—which can become particularly important in wireless sensor networks that involve thousands of tiny, disposable sensor nodes.

The services provided by existing time synchronization methods fall into many disparate points in this parameter space. All of them make tradeoffs—no single method is optimal along all axes.

For example, consumer GPS receivers can synchronize nodes to a persistent-lifetime time standard that is Earth-wide in scope to a precision of 200ns [9]. However, GPS units often can not be used (e.g., inside structures, underwater, during Mars exploration), can require several minutes of settling time. In some cases, GPS units might also be large, high-power and expensive compared to small sensors.

In contrast, consider a small group of nodes with short-range, low-power radios. If one node transmits a signal, the others can use that signal as a time reference—for example, to compare the times at which they recorded a sound. The synchronization provided by this simple "pulse" is local in scope and is limited in precision by the variable delays on the radio receivers and propagation delay of the radio waves. For a given precision bound, the lifetime of the synchronization is also finite as the nodes' clocks will wander after the initial pulse. However, the pulse is fast and energy-efficient because it only requires the transmission of a single signal.

The needs of applications in wireless sensor networks can be characterized along the same axes. For example, consider a beamforming array designed to localize the source of sound, such as that described by YHRCL in [13]. The array described shares a common time base by virtue of the fact that the audio data are all fed to the same processor. For such an array to be implemented on a fully distributed set of autonomous wireless sensors, network time synchronization is needed. This synchronization would require precision of about 100μsec but could be limited in lifetime and local in scope.

Different applications have different synchronization requirements, illustrated by another example: data aggregation. A feature common to sensor networks due to the high energy cost of communication compared to computation [11] is local processing, summarization, and aggregation of data in order to minimize the size and frequency of transmissions. Suppression of duplicate notifications of the same event from a group of nearby sensors can result in significant energy savings [6]. To recognize duplicates, events must be timestamped with a precision on the same order as the event frequency; this might only be tens or hundreds of milliseconds. Since the data may be sent a long way through the network and even cached by many of the intermediate nodes, the synchronization must be broad in scope and long in lifetime—perhaps even persistent.

2.1. Sensor Network Time

A number of factors make existing methods inadequate for timekeeping in a sensor network. Perhaps the most important is that sensor networks must be highly *energy-efficient*. As we mentioned in Section 1, nodes will be untethered and have finite battery reserves. Unlike laptops or other handheld devices that enjoy constant attention and maintenance by humans, the scale of a sensor net's deployment will make replenishment of these reserves impossible. Existing time synchronization methods are not designed with this constraint in mind. Although protocols such as NTP are conservative in their use of bandwidth, they are inefficient in this new context where radios consume significant power even by passively listening for messages [11].

Another complication is introduced by the heterogeneity of hardware that may be used within a sensor network. The smallest nodes—perhaps designed to be attached directly to the phenomena that they are monitoring—are unlikely to have an energy budget or form factor that allows anything more than a local oscillator and a short-range radio. Some will be better endowed, with longer-range radios capable of synchronizing with more remote parts of the network, or in some cases with external time sources such as GPS or WWVB.

The heterogeneity in the synchronization requirements across sensor network applications, the need for energy-efficiency and other constraints not found in conventional distributed systems, and the variety of hardware on which sensor networks will be deployed, all lead us to the following conclusions:

1. The time synchronization methods used by existing distributed systems are not appropriate in sensor networks without modification.

2. Because it is impossible for any *single* synchronization method to appropriate in all situations, sensors should have multiple methods available. If a node can dynamically trade precision for energy, or scope for convergence time, it can avoid "paying" for something that it doesn't need. Ideally, the algorithms should also be *tunable*—allowing finer control over an algorithm than simply turning it on or off.

We are therefore extending a range of traditional ways of synchronizing time for sensor networks. By modifying existing methods and composing them into multi-modal solutions, we can create new forms of synchronization that cover a variety of points in the parameter space we described earlier. Our goal is to implement and characterize a set of methods rich enough so that all applications will have one available that is both necessary and sufficient for its needs.

Starting down this path, we have developed a technique called *post-facto synchronization* to reconcile the need of many applications for accurate sensor event timestamps with the desire to keep the node off in order to conserve energy.

3. Post-Facto Synchronization

To save energy in a sensor network, it is a desirable to keep nodes in a low-power state, if not turned off completely, for as long as possible. Sensor network hardware is often designed with this goal in mind; processor have various "sleep" modes or are capable of powering down high-energy peripherals when not in use.

This type of design is exemplified by the WINS platform [1], which has an extremely low-power "pre-processor" that is capable of rudimentary signal processing. Normally, the entire node is powered down except for the pre-processor. When the pre-processor detects a potentially interesting signal, it powers on the general purpose processor for further analysis. The CPU, in turn, can power on the node's radio if it determines that an event has occurred that needs to be reported.

Such designs allow the components that consume the most energy to be powered for the least time, but also pose significant problems if we wish to keep synchronized time. Traditional methods try to keep the clock disciplined at all times so that an accurate timestamp is always available. What happens if the radio—our external source of time and frequency standards—is continuously off for hours at a time? Or, in the case of a platform like WINS, what if the general-purpose processor that knows how to discipline the clock is also off?

Our solution to this problem is *post-facto synchronization*. In our scheme, nodes' clocks are normally *un*synchronized. When a stimulus arrives, each node records the time of the stimulus with respect to its own local clock. Immediately afterwards, a "third party" node—acting as a beacon—broadcasts a synchronization pulse to all nodes in the area using its radio. Nodes that receive this pulse use it as an instantaneous time reference and can normalize their stimulus timestamps with respect to that reference.

This kind of synchronization is not applicable in all situations, of course: it is limited in scope to the transmit range of the beacon and creates only an "instant" of synchronized time. This makes it inappropriate for an application that needs to communicate a timestamp over long distances or times. However, it does provide exactly the service necessary for beam-forming applications, localization systems, and other situations in which we need to compare the relative arrival times of a signal at a set of spatially local detectors.

3.1. Expected Sources of Error

There are three main factors that affect the accuracy and precision achievable by post-facto synchronization. Roughly in order of importance, they are: receiver clock skew, variable delays in the receivers, and propagation delay of the synchronization pulse.

- **Skew in the receivers' local clocks.** Post-facto synchronization requires that each receiver accurately measure the interval that elapses between their detection of the event and the arrival of the synchronization pulse. However, nodes' clocks do not run at exactly the same rate, causing error in that measurement. Since clock skew among the group will cause the achievable precision to decay as time elapses between the stimulus and pulse, it is important to minimize this interval.

  One way of reducing this error is to use NTP to discipline the frequency of each node's oscillator. This exemplifies our idea of multi-modal synchronization. Although running NTP "full-time" defeats one of the original goals of keeping the main processor or radio off, it can still be useful for frequency discipline (much more so than for phase correction) at very low duty cycles.

- **Variable delays on the receivers.** Even if the synchronization signal arrives at the same instant at all receivers, there is no guarantee that each receiver will *detect* the signal at the same instant. Nondeterminism in the detection hardware and operating system issues such as variable interrupt latency can contribute unpredictable delays that are inconsistent across receivers. The detection of the event itself (audio, seismic pulses, etc.) may also have nondeterministic delays associated with it. These delays will contribute directly to the synchronization error.

  Our design avoids error due to variable delays in the *sender* by considering the sender of the sync pulse to be a "third party." That is, the receivers are considered to be synchronized only with each other, not with the beacon.

  It is interesting to note that the error caused by variable delay is the same irrespective of the time elapsed between the event and the sync pulse. This is in contrast to error due to clock skew that grows over time.

- **Propagation delay of the synchronization pulse.** Our method assumes that the synchronization pulse is an absolute time reference at the instant of its arrival—that is, that it arrives at every node at exactly the same time. In reality, this is not the case due to the finite propagation speed of RF signals. Synchronization will never be achievable with a precision better than the difference in the propagation delay between the various receivers and the synchronization beacon.

  This source of error makes our technique most useful when comparing arrival times of phenomena that propagate much more slowly than RF, such as audio. The six-order-of-magnitude difference in the speed of RF and audio has been similarly exploited in the past in systems such as the ORL's Active Bat [12] and Girod's acoustic rangefinder [5].

3.2. Empirical Study

We designed an experiment to characterize the performance of our post-facto synchronization scheme. The experiment attempts to measure the sources of error described in the previous section by delivering a stimulus to each receiver at the same instant, and asking the receivers to timestamp the arrival time of that stimulus with respect to a synchronization pulse delivered via the same mechanism. Ideally, if there are no variable delays in the receivers or skew among the receivers' local oscillators, the times reported for the stimulus should be identical. In reality, these sources of error cause the dispersion among the reported times to grow as more time elapses between the stimulus and the sync pulse. The decay in precision should happen more slowly if NTP is simultaneously used to discipline the frequency of the receivers' oscillators.

We realized this experiment with one sender and ten receivers, each of which was ordinary PC hardware (Dell OptiPlex GX1 workstations) running the RedHat Linux operating system. Each stimulus and sync pulse was a simple TTL logic signal sent and received by the standard PC parallel port.[1] In each trial, each receiver reported its perceived elapsed time between the stimulus and synchronization pulse according to the system clock, which has 1μsec resolution. We defined the dispersion to be the standard deviation from the mean of these reported values. To minimize the variable delay introduced by the operating system, the times of the incoming pulses were recorded by the parallel port interrupt handler using a Linux kernel module.

In order to understand how dispersion is affected by the time elapsed between stimulus and sync pulse, we tested the dispersion for 21 different values of this elapsed time, ranging from $2^4\mu$sec to $2^{24}\mu$sec (16μsec to 16.8 seconds). For each elapsed-time value, we performed 50 trials and reported the mean. These 1,050 trials were performed in a random order over the course of one hour to minimize the effects of systematic error (e.g. changes in network activity that affect interrupt latency).

For comparison, this entire experiment was performed in three different configurations:

1. The experiment was run on the "raw clock": that is, while the receivers' clocks were not disciplined by any external frequency standard.

2. An NTPv3 client was started on each receiver and allowed to synchronize (via Ethernet) to our lab's stratum-1 GPS clock for ten days. The experiment was then repeated while NTP was running.

3. NTP's external time source was removed, and the NTP daemon was allowed to free-run for several days using its last-known estimates of the local clock's frequency. The experiment was then repeated.

To compare our post-facto method to the precision achievable by NTP alone, we recorded two different stimulus-arrival timestamps when running the experiment in Configuration 2: the time with respect to the sync pulse *and* the time according to the NTP-disciplined local clock. Similar to the other configurations, a dispersion value for NTP was computed for each stimulus by computing the standard deviation from the mean of the reported timestamps. The horizontal line in Figure 1 is the mean of those 1,050 dispersion values—101.70 μsec.

Our results are shown in Figure 1. [2]

[1] This was accomplished using the author's parallel port pin programming library for Linux, `parapin`, which is freely available at http://www.circlemud.org/jelson/software/parapin

[2] "The joy of engineering is to find a straight line on a double logarithmic diagram." –Thomas Koenig

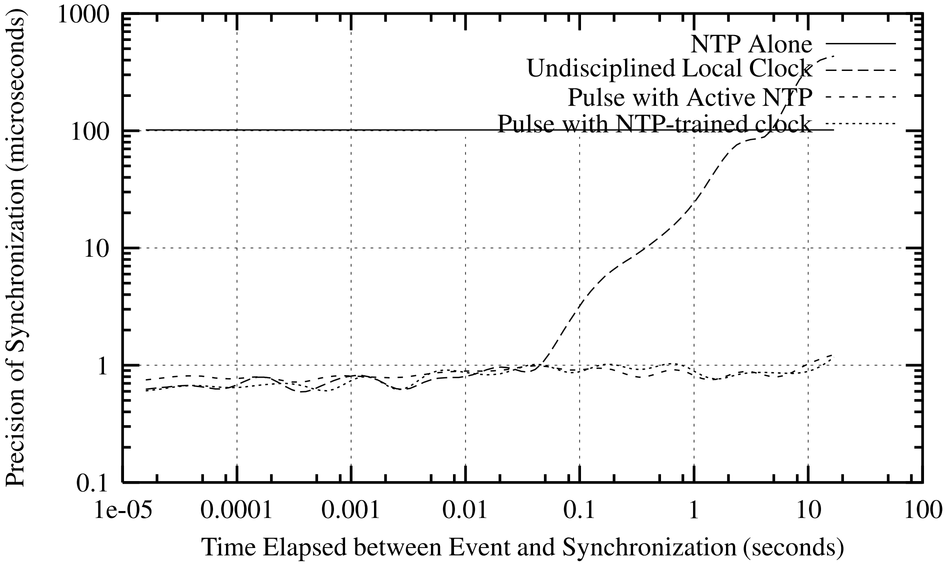

Figure 1. Precision of post-facto time synchronization without external frequency discipline, with discipline from an active NTP time source, and with free-running NTP discipline (external time source removed after the oscillator's frequency was estimated). These are compared to the precision achievable with NTP alone (the horizontal line near 100μsec). The breakpoint seen near 50msec is where error due to clock skew, which grows proportionally with the time elapsed from stimulus to sync pulse, overcomes other sources of error that are independent of this interval. Each point represents the dispersion experienced among 10 receivers, averaged over 50 trials.

3.3. Discussion

The results shown in Figure 1 illuminate a number of aspects of the system. First, the experiment gives insight into the nature of its error sources. The results with NTP-disciplined clock case are equivalent to undisciplined clocks when the interval is less than $\approx$ 50msec, suggesting that the primary source of error in these cases is variable delays on the receiver (for example, due to interrupt latency or the sampling rate in the analog-to-digital conversion hardware in the PC parallel port). Beyond 50msec, the two experiments diverge, suggesting that clock skew becomes the primary source of error at this point.

Overall, the performance of post-facto synchronization was quite good. When NTP was used to discipline the local oscillator's frequency, precision very near to the clock's resolution of 1μsec was achieved. This is significantly better than the 100μsec achieved by NTP alone. Clearly, the combination of NTP's frequency estimation with the sync pulse's instantaneous phase correction was very effective. Indeed, the multi-modal combination achieves precision better than either mode can achieve alone. We find this a very encouraging indicator for the multi-modal synchronization framework we proposed at the end of Section 2.

Without NTP discipline, the post-facto method still performs reasonably well for short intervals between stimulus and sync pulse. For longer intervals, we are at the mercy of happenstance: the precision depends on the natural frequencies of whatever oscillators we happen to have in our receiver set.

Perhaps the most exciting result, however, is shown in the experiment where NTP disciplined the nodes' local clocks using only the last-known-estimate of frequency, after the external time source was removed. The achievable precision was 1μsec: the limit of our clock's resolution and, more importantly, exactly the same as the result with NTP and an active external time standard. This result is important because it shows that extremely low-energy and high-precision time synchronization is possible: after an initial frequency-training period, nodes might be able to keep their radios off for days and still instantly acquire a 1μsec-precision timebase when an event of interest arrives. That result is made possible by the multi-modal synchronization; the frequency correction provided by free-running NTP is not good enough to keep clocks precisely in phase over time due to accumulated error. (In the free-running NTP experiment, the accuracy of the timestamps when not normalized by the sync pulse was only in the tens of milliseconds.)

All of these results, while encouraging, do come with a number of caveats. Our experiments results were performed under idealized laboratory conditions, using (equal-length) cables to directly connect the sender to the receivers. Real world conditions will require wireless links that are likely far more complex with more opportunities for variable delay. In addition, the relatively constant ambient temperature reduced the oscillators' frequency drift over time. A real sensor network deployed outdoors might not be able let NTP free-run without an external time source for as long as we did in our experiment.

4. Future Work

As part of our continuing research, we plan to re-implement our time synchronization experiment on hardware that is more akin to hardware that will be found in sensor networks: slower, lower-power nodes that have wireless radio links. While using PCs with wired stimuli and event delivery did provide an important proof-of-concept, we wish to investigate the effects on precision of factors such as slower clock speeds, variable latency of radios, and nondeterminism introduced by radio propagation anomalies. We plan to do these tests using the wireless sensor network testbed in place as part of the related SCADDS project [4].

In addition to characterizing post-facto synchronization, we plan to use it in the context of a real application: localization. Building on Girod's prototype acoustic rangefinder [5], we plan to use post-facto time synchronization to facilitate measurement of the time of flight of sound from an audio source to a set of receivers, allowing them to trilaterate with high precision.

We also plan to build on this work by developing additional time sync methods with the ultimate goal of providing a palette to applications that covers a good portion of the parameter space we described in of the parameter space we described in Section 2. Because it is impossible for any *single* synchronization method to appropriate in all situations, sensors should have multiple methods available. If a node can dynamically trade precision for energy, or scope for convergence time, it can avoid "paying" for something that it doesn't need. Ideally, the algorithms should also be *tunable*—allowing finer control over an algorithm than simply turning it on or off.

5. Conclusions

Time synchronization is a critical piece of infrastructure for any distributed system. Distributed, wireless sensor networks make heavy use of synchronized time, but often have unique requirements in the scope, lifetime, and precision of the synchronization achieved, as well as the time and energy required to achieve it. Existing time synchronization methods need to be extended to meet these new needs.

We have presented an implementation of our own sensor network time synchronization scheme, *post-facto synchronization*. This method combines the oscillator frequency discipline provided by NTP with an instantaneous phase correction provided by a simple synchronization signal sent by a beacon. Our experiments have shown achievable timing precision for a group of 10 nodes to be at the limit of our clock resolution of 1μsec.

An important additional result is that the same timing precision was possible even when NTP no longer had an active external time or frequency standard, after an initialization period when it was allowed to estimate the local oscillator's frequency error. This is critical for sensor networks where limited energy reserves and the high energy cost of operating a wireless radio make standard NTP unsuitable for long-lived, low-power operation.

Although our current results are a preliminary laboratory study, we believe that post-facto synchronization over wireless radios will be able to support the same instantaneous creation of a short-lived but highly precise synchronized timebase ever after a long period of radio silence. Our ongoing research is moving our experiment from the lab to real sensor network nodes where we plan to characterize our scheme with further experiments and use it in the context of real applications.

Acknowledgments

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. Lin, F. Newberg, G. Pottie, W. Kaiser, and H. Marcy. Wireless Integrated Network Sensors: Low Power Systems on a Chip. In *Proceedings of the European Solid State Circuits Conference*, 1998.

[2] R. Beehler. Time/frequency services of the U.S. National Bureau of Standards and some alternatives for future improvement. *Journal of Electronics and Telecommunications Engineers*, 27:389–402, Jan 1981.

[3] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat monitoring: Application driver for wireless communications technology. In *Proceedings of the 2001 ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, April 2001.

[4] D. Estrin, R. Govindan, J. Heidemann, and S. 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.

[5] L. Girod. Development and characterization of an acoustic rangefinder. Technical Report 00-728, University of Southern California, Department of Computer Science, March 2000.

[6] C. Intanagonwiwat, R. Govindan, and D. 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, Aug. 2000. ACM Press.

[7] J. Kahn, R. Katz, and K. 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.

[8] E. D. Kaplan, editor. *Understanding GPS: Principles and Applications*. Artech House, 1996.

[9] J. Mannermaa, K. Kalliomaki, T. Mansten, and S. Turunen. Timing performance of various GPS receivers. In *Proceedings of the 1999 Joint Meeting of the European Frequency and Time Forum and the IEEE International Frequency Control Symposium*, pages 287–290, April 1999.

[10] D. L. Mills. Internet Time Synchronization: The Network Time Protocol. In Z. Yang and T. A. Marsland, editors, *Global States and Time in Distributed Systems*. IEEE Computer Society Press, 1994.

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

[12] A. Ward, A. Jones, and A. Hopper. A new location technique for the active office. *IEEE Personal Communications*, 4(5):42–47, October 1997.

[13] K. Yao, R. Hudson, C. Reed, D. Chen, and F. Lorenzelli. Blind beamforming on a randomly distributed sensor array system. *IEEE Journal of Selected Areas in Communications*, 16(8):1555–1567, Oct 1998.