Tracking Real-World Phenomena with Smart Dust

Kay Römer

Institute for Pervasive Computing
Dept. of Computer Science
ETH Zurich, Switzerland
roemer@inf.ethz.ch

Abstract. So-called "Smart Dust" is envisioned to combine sensing, computing, and wireless communication capabilities in an autonomous, dust-grain-sized device. Dense networks of Smart Dust should then be able to unobtrusively monitor real-world processes with unprecedented quality and scale. In this paper, we present and evaluate a prototype implementation of a system for tracking the location of real-world phenomena (using a toy car as an example) with Smart Dust. The system includes novel techniques for node localization, time synchronization, and for message ordering specifically tailored for large networks of tiny Smart Dust devices. We also point out why more traditional approaches developed for early macro prototypes of Smart Dust (such as the Berkeley Motes) are not well suited for systems based on true Smart Dust.

1 Introduction

Smart Dust is commonly used as a synonym for tiny devices that combine sensing, computing, wireless communication capabilities, and autonomous power supply within a volume of only few cubic millimeters at low cost. The small size and low per-device cost allows an unobtrusive deployment of large and dense Smart Dust populations in the physical environment, thus enabling detailed in-situ monitoring of real-world phenomena, while only marginally disturbing the observed physical processes. Smart Dust is envisioned to be used in a wide variety of application domains, including environmental protection (identification and monitoring of pollutions), habitat monitoring (observing the behavior of animals in their natural habitats), and military systems (monitoring activities in inaccessible areas). Due to its tiny size, Smart Dust is expected to enable a number of novel applications. For example, it is anticipated that Smart Dust nodes can be moved by winds or can even remain suspended in air, thus supporting better monitoring of weather conditions, air quality, and many other phenomena. Also, it is hard to detect the bare presence of Smart Dust and it is even harder to get rid of it once deployed, which might be helpful for many sensitive application areas.

Current research (cf. [1] for an overview) is mainly focusing on so-called COTS (Commercial Off The Shelf) Dust, early macro prototypes of Smart Dust. COTS Dust nodes such as the Motes [24] developed at UC Berkeley are built from commercially available hardware components and still have a volume of several cubic centimeters. Unfortunately, these devices cannot be simply scaled down to the cubic millimeter size of true Smart Dust. First Smart Dust prototypes [21] demonstrate that the tremendous

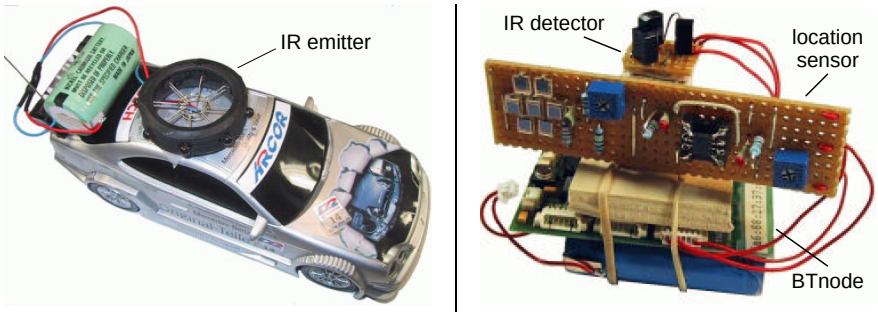
volume reduction (factor 1000 and more) may require radical changes in the employed technologies (e.g., use of optical instead of radio communication) compared to COTS Dust. These technological changes have important implications for algorithms, protocols, systems, and infrastructure. Our goal is to examine these implications and develop solutions for the resulting problems. To identify and illustrate these issues, we have developed an object tracking system that makes use of Smart Dust. This system allows tracking the location of "targets" with Smart Dust, using a remote-controlled toy car as a sample target.

Since Smart Dust hardware is currently in a very early prototyping stadium, our implementation still uses COTS Dust. However, our algorithms and protocols are designed to be directly portable to true Smart Dust, once the hardware becomes available. In the following sections we first outline the characteristics of Smart Dust, before presenting the details of our object tracking system. Particularly, this includes novel techniques for synchronizing time among the nodes of the network, for localizing Smart Dust nodes in physical space, and for ordering event notifications according to their time of occurrence. This will be followed by the presentation of some measurements and a discussion.

2 Smart Dust

The envisioned dust-grain size of Smart Dust nodes has a number of important implications with respect to their hardware design. Most importantly, it is hardly possible to fit current radio communication technology into Smart Dust – both size-wise (antennas) and energy-wise (power consumption of current transceivers) [21]. Hence, Smart Dust prototypes developed at UC Berkeley utilize a more power and size efficient passive laser-based communication scheme to establish a bidirectional communication link between dust nodes and a so-called base station transceiver (BST). For downlink communication (BST to dust), the base station points a modulated laser beam at a node. The latter uses a simple optical receiver to decode the incoming message. For uplink communication (dust to BST), the base station points an unmodulated laser beam at a node, which in turn modulates and reflects back the beam to the BST. For this, the dust nodes are equipped with a so-called Corner Cube Retro Reflector (CCR), which is a special Micro Electro-Mechanical System (MEMS) structure consisting of three mutually perpendicular mirrors. The CCR has the property that any incident ray of light is reflected back to the source under certain conditions. If one of the mirrors is misaligned, this retroreflection property is spoiled. The Smart Dust CCR includes an electrostatic actuator that can deflect one of the mirrors at kilohertz rates. Using this actuator, the incident laser beam is "on-off" modulated and reflected back to the BST.

This type of design implies a single-hop network topology, where dust nodes cannot directly communicate with each other, but only with a base station. The base station can be placed quite far away from the nodes, since the employed laser communication works over a range of hundreds of meters, provided a free line-of-sight between the BST and the nodes. Communication may suffer from significant and highly variable delays if the laser beam is not already pointing at a node which is subject to communication



(a) (b)

Fig. 1. (a) Remote-controlled car (b) "Smart Dust" node.

with the BST. Smart Dust nodes can be highly mobile, since nodes are small enough to be moved by winds or even to remain suspended in air, buoyed by air currents.

Early prototypes of Smart Dust [22] implement the optical receiver, CCR, a light sensor, a solar cell, and a simple control unit within 16 cubic millimeters. Future devices are expected to include a complete processing unit instead of the simple control unit, provide a wider variety of sensors, and will feature further reductions in size and energy consumption.

Due to the limited size, on-board devices for energy storage and harvesting can only provide a tiny amount of energy, thus every hardware and software component has to be optimized for energy efficiency. Despite the passive optical communication scheme, data communication still consumes most of the energy. Hence, communication has to be kept to a minimum. In addition, many thousands of sensors may have to be deployed for a given task – an individual sensor's small effective range relative to a large area of interest makes this a requirement, and its small form factor and low cost makes this possible. Therefore, scalability is another critical factor in the design of the system. Note that Smart Dust is subject to frequent communication failures (e.g., line-of-sight obstructions) and node failures (e.g., destruction due to environmental influences, depleted batteries). Hence, applications must be robust against these types of failures.

While the tiny size of Smart Dust nodes leads to a number of challenging problems as pointed out above, the anticipated high density of Smart Dust deployments may allow to monitor environmental phenomena with unprecedented quality and detail. The tracking system described in the following section has been specifically designed with the above requirements and characteristics in mind.

3 The Tracking System

The purpose of the tracking system is to track the location of real-world phenomena with a network of Smart Dust nodes. We use a remote-controlled toy car (Figure 1 (a)) as a sample target. The current tracking system assumes that there is only one car. Wirelesse sensor nodes are randomly deployed in the area of interest and can change their location after deployment. When they detect the presence of the car (Section 3.1),

they send notifications to a base station. The base station fuses these notifications (Section 3.2) in order to estimate the current location of the car. A graphical user interface displays the track and allows to control various aspects of the system. The data fusion process requires that all nodes share a common reference system both in time and space, which necessitates mechanisms for node localization (Section 3.3) and time synchronization (Section 3.4).

Unfortunately, true Smart Dust hardware is not yet available. Some recent prototypes are presented in [22], but they still do not include important components like a full-fledged processing unit. Therefore, we still use COTS Dust for the implementation of our prototype system. Note however, that this is only an intermediate step to demonstrate the feasibility of our approaches. Our ultimate goal is to implement the tracking system using true Smart Dust.

The current sensor node hardware is based on BTnodes (Figure 1 (b)) [25], which provide Bluetooth communication, an ATMEL ATMega 128L microcontroller, and various external interfaces for attaching sensors on a matchbox-sized printed circuit board. The base station consists of a Linux laptop computer equipped with a Bluetooth radio. In analogy to the single-hop network topology of Smart Dust described in Section 2, BTnodes do not directly communicate with each other, but only with the base station. Before communication can take place, the base station has to set up a so-called Bluetooth Piconet containing no more than 7 BTnodes. To support more than 7 nodes, the base station has to periodically switch the Piconet in a round robin fashion, such that eventually every BTnode gets a chance to talk to the base station. Again, this is very similar to Smart Dust, where the base station has to point the (typically slightly defocused) laser beam at a group of nodes in order to enable communication with them.

3.1 Target Detection

Tracking targets with networks of sensors has been an active research topic for many years, [3, 8, 18] give a good overview of many tracking algorithms. Most of the approaches are optimized for sparse networks, where a high tracking accuracy should be achieved despite a relatively low node density. To achieve this, many approaches make a number of assumptions about the tracked target. Methods which estimate target distance based on signal strength estimates, for example, require knowledge of the intensity of the signal emitted by the target in order to achieve good accuracy. Approaches based on measuring the difference of time of arrival of a signal emitted by the target at different sensor nodes are typically limited to sound or other signal modalities with low propagation speed. Signal modalities with high propagation speeds such as radio waves would require distributed clock synchronization with an accuracy of few nanoseconds, which is typically not available. Other approaches need to know lower and upper bounds of the velocity or the acceleration of the tracked target.

While these assumptions help to achieve good tracking accuracy, they also limit the applicability of the tracking system. In order to make our system applicable to a wide variety of targets, we tried to avoid making assumptions about the target as much as possible. In order to achieve a satisfactory tracking accuracy nevertheless, we exploit the anticipated high density of Smart Dust deployments – which is expected because of the intended small size and low cost of Smart Dust devices.

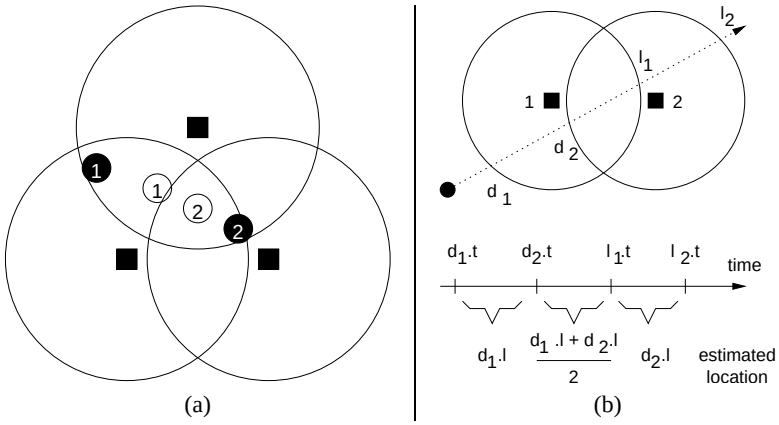


Fig. 2. (a) Estimating car location by centroids (b) Data fusion algorithm.

Our approach assumes that the presence of the target can be detected with an omnidirectional sensor featuring an arbitrary but fixed sensing range r, that is, the sensor can "see" the target if and only if the distance to the target is lower than r. The data fusion algorithm presented in the following section needs to know an upper bound R of this sensing range. In many applications, the target cannot be instrumented for tracking purposes (e.g., a cloud of toxic gas, an oil slick, fire). The remote-controlled car (used as a sample target in our tracking system) emits a characteristic acoustic signature which could be used for detection. However, this signature depends on the velocity of the car. To avoid the intricacies with detecting this variable signature, we chose in our experiment a different solution based on infrared (IR) light, leaving detection based on the car's acoustic signature as future work.

In the current version of the prototype, we equipped the car with an omnidirectional IR light emitter consisting of eight IR LEDs mounted on top of the car (Figure 1 (a)). Accordingly, the sensor nodes are equipped with an omnidirectional IR light detector consisting of three IR photo diodes (Figure 1 (b)). The used IR photo diodes include a filter to cancel out visible light. The output of the IR detector is connected to an analog-to-digital (ADC) converter of the BTnode's microcontroller. If the output value of the ADC exceeds a certain threshold, the presence of the car is assumed. Using a low-pass filter, the threshold value is adopted to slowly changing daylight, which also contains IR components. With this setup, the BTnodes can detect the car at a distance of up to approximately half a meter. When a node first detects the car, it sends a "detection notification" to the base station, containing its node ID as well as its time and location at the time of detection. When the node no longer sees the car, it sends a "loss notification" to the base station, which contains its node ID and its current time. If the node changes its location during the presence of the car, a loss notification is emitted, followed by a detection notification with the new node location.

3.2 Data Fusion

Following the argumentation at the beginning of the previous section, we try to avoid making assumptions about the target as much as possible. Therefore, the base station has to derive the current location of the tracked target solely based on detection notifications and loss notifications received from the sensor nodes.

We use an approach that estimates the car's location at time t by the centroid of the locations of the sensor nodes that "see" the car at time t. The centroid of a set of N locations $\{l_i = (x_i, y_i, z_i)\}$ is defined as $\hat{l} := \frac{1}{N}\sum l_i = \left(\frac{1}{N}\sum x_i, \frac{1}{N}\sum y_i, \frac{1}{N}\sum z_i\right)$. Consider Figure 2 (a) for an example. Depicted are three sensor nodes (black squares) with their respective sensing ranges, and two car locations (black circles). The hollow circles indicate the respective estimated locations (i.e., centroids).

Figure 2 (b) illustrates an algorithm to calculate the car location estimates given the detection and loss notifications received from the sensor nodes as described in Section 3.1. The figure shows sensor nodes 1 and 2, their respective sensing ranges, and the trajectory of the car (dotted arrow). When the car enters the sensing range of node i, a detection notification d_i is emitted, containing time $d_i.t$ and location $d_i.l$ of node i at the time of detection. Accordingly, node i emits a loss notification l_i when the car leaves the sensing range. In a first step, all notifications are sorted by increasing timestamps $d_i.t$ ($l_i.t$) as depicted on the time axis in the lower half of Figure 2 (b). In a second step, we iterate over these sorted notifications from left to right, recording the active nodes (those that currently see the car) in a set S. If we come across a loss notification l_i, we remove i from S. If we come across a detection message d_i, we add i to S. Additionally, we remove all nodes j from S, whose sensing ranges do not overlap with the detection range of node i, that is, for which $|d_i.l - d_j.l| > 2R$ holds. This is necessary to compensate for missed loss notifications, which would otherwise permanently affect the accuracy of the tracking system by not removing the respective entries from S. A missing enter notification will lead to a temporarily decreased tracking accuracy, but will not otherwise permanently affect the system.

The location of the car during the time interval starting at the currently considered notification and ending at the next notification is estimated by the centroid $\hat{S}$ of the locations of the nodes in S (i.e., $d_1.l$ during $[d_1.t, d_2.t)$, $(d_1.l + d_2.l)/2$ during $[d_2.t, l_2.t)$, and $d_2.l$ during $[l_1.t, l_2.t)$).

The localization accuracy of a similar centroid-based algorithm was examined in [2] in a different context under the assumption that nodes are located on a regular grid. We can interpret their results for our setting as follows. The localization accuracy depends on the sensing range r of the nodes (about 50cm in our case) and the distance d between adjacent nodes. For $r/d = 2$ (i.e., $d \approx$ 25cm in our case) the average and maximum localization errors are $0.2d$ (i.e., 5cm) and $0.5d$ (i.e., 12.5cm), respectively. In general, larger r/d values yield better accuracy. Therefore, the accuracy can be improved by increasing the node deployment density, since that reduces d while keeping r constant.

3.3 Node Localization

In order to derive the location of the tracked car from proximity detections as described in Section 3.1, the locations of the sensor nodes have to be estimated. Various systems

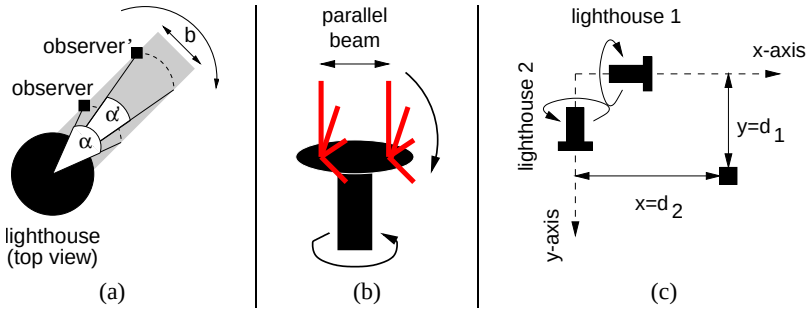


Fig. 3. (a) Lighthouse with parallel beam (b) Lighthouse implementation with two rotating laser beams (c) 2D location system using two lighthouses.

have been designed for localization purposes. However, most of them are not suited for Smart Dust. Will will first explain why this is the case, before outlining the solution we developed for the localization of Smart Dust nodes.

Energy, size, and cost constraints preclude equipping Smart Dust with receivers for localization infrastructures like GPS. With Smart Dust, it might not even be possible to equip sensor nodes with transceivers for radio waves or ultra sound due to the tiny size and energy budget of Smart Dust nodes. Hence, traditional ranging approaches such as ones based on time of flight of ultrasound signals or received radio signal strength might be unusable in the context of Smart Dust.

Many localization systems such as [2, 20] depend on an extensive hardware infrastructure. Localization systems based on trilateration, for example, require many spatially distributed and well-placed infrastructure components in order to achieve high accuracy. This is not an adequate solution for Smart Dust, since it contradicts the ad hoc nature of Smart Dust, where nodes may have to be deployed in remote, inaccessible, or unexploited regions. Other localization approaches such as [4, 16] require centralized computation, which results in systems that do not scale well to large numbers of nodes. To overcome the limitations of infrastructure-based approaches, various schemes for ad hoc localization have been devised (e.g., [14, 15]). However, these schemes depend on inter-node communication, which is not scalable with Smart Dust, since any inter-node communication has to pass through the BST.

An important overhead involved in setting up a localization system is node calibration in order to enforce a correct mapping of sensor readings to location estimates [23]. In systems based on radio signal strength, for example, the received signal strength is mapped to a range estimate. Variations in transmit power and frequency among the nodes can cause significant inaccuracies in the range estimates when used without calibration. Since the cheap low-power hardware used in sensor nodes typically introduces a high variability between nodes, sensor nodes have to be individually calibrated. This, however, may not be feasible in large networks.

To overcome the above limitations, we designed and implemented a novel localization system for Smart Dust. This system consists of a single infrastructure device, which emits certain laser light patterns. By observing these patterns, dust nodes can autonomously estimate their location with high accuracy. Since dust nodes only pas-

sively observe light flashes, this system is very energy efficient on the side of the nodes. Moreover, optical receivers consume only little power and can be made small enough to fit in a volume of few cubic millimeters. Since dust nodes do not need to interact with other nodes in order to estimate their location, the system scales to very large networks. Also, node calibration is not necessary due to using differential measurements, where constant offsets cancel out due to using the difference between two measurements that use the same signal path.

To understand our localization approach, consider a lighthouse with a *parallel* beam (i.e., a beam with constant width b) as depicted in Figure 3 (a). Assume that it takes the lighthouse t_{turn} for one rotation. When the parallel beam passes by an observer (black square), the observer will see the lighthouse flash for a certain period of time t_{beam}. Note that t_{beam} and hence the angle $\alpha = 2\pi t_{\mathrm{beam}}/t_{\mathrm{turn}}$ under which the observer sees the beam, depend on the observer's distance d from the lighthouse rotation axis, since the beam is parallel. Using α, the observer's distance d from the lighthouse rotation axis can be expressed as $b/(2\sin(\alpha/2))$.

A parallel beam can be implemented as depicted in Figure 3 (b): two rotating laser beams (at high speeds) define the outline of a wide "virtual" parallel beam, which in turn is rotating around a central axis (at much lower speeds) to create a rotating lighthouse effect. An observer looking at such a lighthouse sees two sequences of short laser flashes as the two "laser light planes" rotate by. t_{beam} can then be obtained by measuring the amount of time elapsed between the two flash sequences.

Using two such lighthouses, a 2D location system can be constructed as depicted in Figure 3 (c). The two lighthouses are assembled such that their rotation axes are mutually perpendicular. The distances d_1 and d_2 to the lighthouse rotation axes then equal the y and x coordinates of the observer in the 2-dimensional coordinate system defined by the lighthouse rotation axes. Accordingly, a 3D location system can be built out of 3 lighthouses with mutually perpendicular rotation axes. Realizing a lighthouse with an exactly parallel beam is very difficult in practice. Therefore we developed a geometrical model of a more realistic lighthouse with an approximately parallel beam in [13]. Using this model, a 2D prototype (see Figure 4) of the system allows to estimate sensor node locations with an accuracy of about 5cm in our tracking system.

Figure 1 (b) shows a BTnode with the hardware for receiving the laser light flashes attached to it. The size of the current prototype of this receiver hardware is about 7cm $\times$ 3cm. Using an ASIC, however, this receiver can be fit within few cubic millimeter. The Smart Dust prototypes presented in [22], for example, contain a similar optical receiver within a volume of 16 cubic millimeters.

3.4 Time Synchronization

The car location estimation described in Section 3.2 assumes that the timestamps contained in notification messages refer to a common physical time scale, requiring synchronization of clocks of the Smart Dust nodes. Various clock synchronization schemes exist for a variety of application domains. We will first outline why these approaches are not suitable for Smart Dust, before presenting our own synchronization approach.

As with node localization, energy, size, and cost constraints preclude equipping Smart Dust with receivers for time infrastructure like GPS [6] or DCF77 [26]. Also,

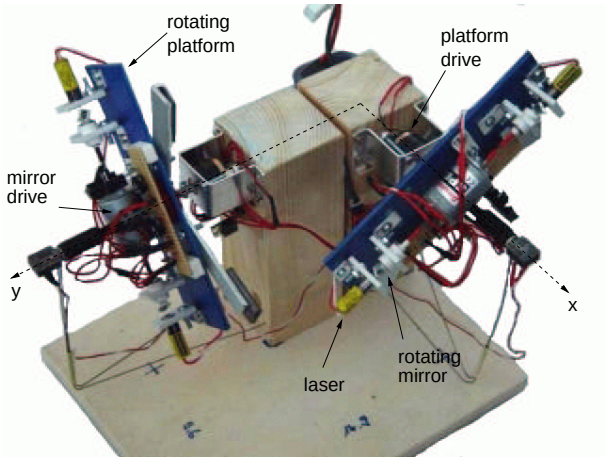


Fig. 4. Lighthouse prototype device for node localization.

logical time [7] is not sufficient, since it only captures causal relationships between "in system" events, defined by message exchanges between event-generating processes. In contrast, phenomena sensed by dust nodes are triggered by external physical events which are not defined by in-system message exchanges; physical time must be used to relate events in the physical world.

Many traditional synchronization approaches such as NTP [10] are based on frequently exchanging "time beacons" – messages containing clock readings of the sender at the time of message generation – among nodes to be synchronized. This leads to synchronization algorithms which are typically not energy efficient. For example, the CPU is used continuously to perform frequency disciplining of the oscillator by adding small increments to the system clock. In addition, synchronization beacons are frequently exchanged, which can require significant amounts of energy. Also, the CPU may not be available if the processor is powered down to save energy.

These problems can be solved by rethinking various aspects of a time synchronization service [5]. Energy efficiency, for example, can be significantly improved by exploiting certain characteristics of sensor network applications. As with our tracking system, sensor networks are typically triggered by physical events. Hence, sensor network activity is rather bursty than continuous and rather local than global. This leads to a situation, where synchronized clocks are only required occasionally and only for certain subsets of nodes. One possible way to exploit these characteristics is called post-facto synchronization. There, unsynchronized clocks are used to timestamp sensor events. Only when two timestamps have to be compared by the application, they are reconciled in order to establish a common time scale.

Our time synchronization approach lets the node's clocks run unsynchronized. Timestamps are generated using these unsynchronized clocks. However, if a notification message containing a timestamp is sent to the base station, the contained timestamp is transformed to the receiver's local time. This transformation can often be performed

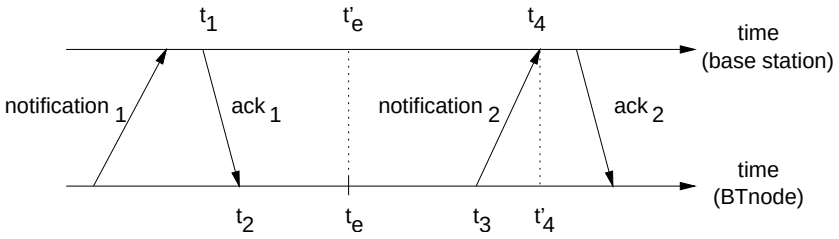


Fig. 5. Time transformation.

without any additional message exchanges by piggybacking on the notification messages that have to be exchanged anyway.

Consider Figure 5 to understand how this can be achieved. The figure depicts two consecutive notification message exchanges (and according acknowledgments) between a sensor node and the base station. At sensor-node-time t_e the car enters the sensing range. The respective notification message is sent to the base station at time t_3, containing t_e as the timestamp. We want to determine t'_e, the base-station time that corresponds to sensor-node-time t_e. If we knew t'_4 (i.e., sensor-node time for base-station-time t_4), we could calculate the message delay $D := t'_4 - t_3$. Using D, we could transform t_e to base-station time by calculating $t'_e := t_4 - D - (t_3 - t_e)$.

Unfortunately we don't know t'_4, since the clocks are unsynchronized. However, we can estimate D by $0 \leq D \leq RTT$ with $RTT := (t_4 - t_1) - (t_3 - t_2)$. This gives us the following estimation for t'_e: $t_4 - RTT - (t_3 - t_e) \leq t'_e \leq t_4 - (t_3 - t_e)$. We can thus transform t_e into the time interval $\tilde{t_e} := [t_4 - RTT - (t_3 - t_e), t_4 - (t_3 - t_e)]$ with $t'_e \in \tilde{t_e}$. We use $\tilde{t_e}$ as the notification's transformed timestamp in the base station. Appropriate interval-arithmetic operators can be used to compare such "time-interval stamps" (e.g., $[t_A^L, t_A^R] < [t_B^L, t_B^R] \Leftrightarrow t_A^R < t_B^L$).

Note that $(t_3 - t_2)$ and $(t_3 - t_e)$ can be piggybacked on the $\mathrm{notification}_2$ message, such that the base station can perform the time transformation without additional message exchanges. Also note that the base station can be relieved of storing t_1 between two consecutive message exchanges by sending back t_1 to the sensor node as part of the ack_1 message. By including t_1 in the following $\mathrm{notification}_2$ message, the base station will have t_1 available to calculate the timestamp transformation.

The above transformation rule can be extended to take into account the clock drift, which we demonstrate in [12]. This approach allows us to estimate t'_e with an accuracy of about 10 milliseconds in our tracking system.

3.5 Message Ordering

The data fusion algorithm described in Section 3.2 requires sorting notifications by their timestamps. The time transformation approach described in Section 3.4 enables us to compare and sort timestamps originating from different nodes. However, we still have to ensure that a notification message is not processed by the data fusion algorithm until all earlier notifications arrived at the base station. This is of particular importance

for Smart Dust, since messages are subject to long and variable delays as described in Section 2.

One particularly attractive approach to message ordering is based on the assumption that there is a known maximum network latency Δ. Delaying the evaluation of inbound messages for Δ will ensure that out-of-order messages will arrive during this artificial delay and can be ordered correctly using their timestamps. That is, message ordering can be achieved without any additional message exchanges. The literature discusses a number of variants of this basic approach [9, 11, 17].

However, there is one major drawback of this approach: the assumption of a bounded and known maximum network latency. As discussed earlier, Smart Dust suffers from long and variable network delays. Using a value for Δ which is lower than the actual network latency results in messages being delivered out of order. Using a large value for Δ results in long artificial delays, which unnecessarily decreases the performance of the tracking system.

We therefore introduce a so-called adaptive delaying technique that measures the actual network delay and adapts Δ accordingly. Doing so, it is possible that the estimated Δ is too small and messages would be delivered out of order. Our algorithm detects such late messages and deletes them (i.e., does not deliver them to the application at all). Recall that the data fusion algorithm presented in Section 3.2 was specifically designed to tolerate missing detection and loss notifications. Hence, deleting a message only results in a less accurate track, since then one Smart Dust node less contributes to the estimation of the target location. We argue that this slight decrease of accuracy is acceptable since deleting a message is a rare event, which only occurs at startup or when the maximum network latency increases during operation (i.e., when the value of Δ is lower than the actual maximum network latency). The expected high density of Smart Dust deployments can also compensate this decrease of accuracy. Additionally, our algorithm includes a parameter which can be tuned to trade off tracking latency for tracking accuracy.

Specifically, the adaptive delaying algorithm executed in the BST maintains a variable Δ holding the current delay value, a variable t_{latest} holding the timestamp of the latest message delivered to the application, and a queue which stores messages ordered by increasing timestamps. Initially, Δ is set to some estimate of the maximum network latency, t_{latest} is set to the current time t_{now} in the base station, and the queue is empty.

Upon arrival of a new notification n with timestamp (interval) $n.t$, the actual message delay $d := t_{\text{now}} - n.t^L$ is calculated[1]. Δ is then set to the maximum of Δ and $c \cdot d$. The constant factor c influences the chance of having to delete an out-of-order message and can thus be tuned to trade off tracking latency for tracking accuracy. We use $c = 1.2$ in our prototype. Now we check if $t_{\text{latest}} < n.t^L$ holds, in which case n can still be delivered in order. If so, n is inserted into the queue at the right position according to $n.t$. Otherwise, n is deleted.

The first element n_0 of the queue (i.e., the one with the smallest timestamp) is removed from the queue as soon as the base station's clock (t_{now}) shows a value greater than $n_0.t^R + \Delta$. Now t_{latest} is set to $n_0.t^R$ and n_0 is delivered to the data fusion algorithm.

[1] t^R (t^L) refers to the right (left) end of the time interval t.

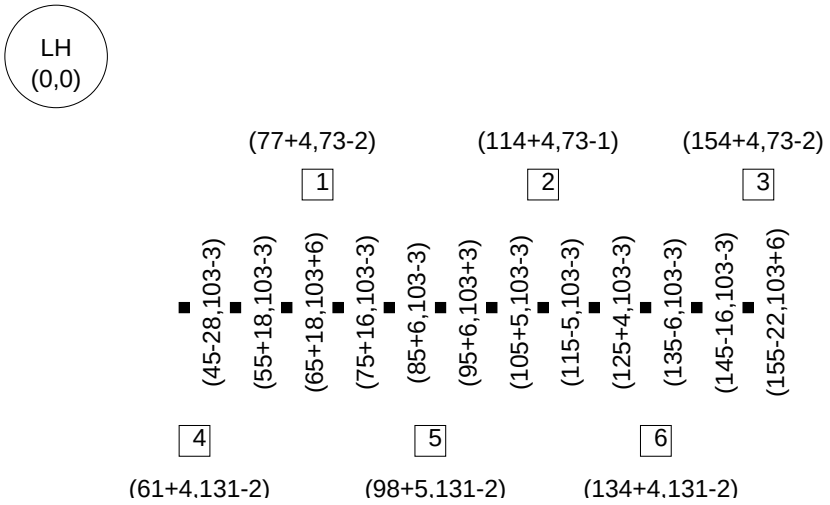


Fig. 6. Measurement setup (not drawn to scale).

4 Evaluation

In order to assess the accuracy of the proposed tracking system, we performed a set of initial measurements. Figure 6 shows the setup of our measurements. The lighthouse device ("LH" in the picture) was placed in the upper left corner and defines the origin (0,0) of a 2D coordinate system. Six sensor nodes (numbered rectangles in the figure) were placed in an area of about one square meter. The car moved then through the sensor field. Location estimates were obtained at 12 positions of the car (indicated by the black squares in the picture). We performed the whole experiment 10 times and calculated averages. The sensor nodes as well as the car locations are annotated with coordinates $(x \pm \Delta_x, y \pm \Delta_y)$, where (x, y) are the ground truth positions in centimeters obtained by a tape measure. $\pm\Delta_x$ and $\pm\Delta_y$ indicate the average errors of the output of the tracking system relative to the ground truth position in the x and y axis, respectively. The *average* error of the sensor node location estimates is $\bar{\Delta}_x = 4.16$cm and $\bar{\Delta}_y = 1.83$cm. We attribute the larger $\bar{\Delta}_x$ value to mechanical problems with one of the lighthouses. The *average* error of the car location estimates is $\bar{\Delta}_x = 12.5$cm and $\bar{\Delta}_y = 3.5$cm. The *maximum* error of the sensor node location estimates is $\hat{\Delta}_x = 5$cm and $\hat{\Delta}_y = 2$cm. The *maximum* error of the car location estimates is $\hat{\Delta}_x = 28$cm and $\hat{\Delta}_y = 6$cm. The difference between the values for the x and y axis is due to the asymmetry of the node arrangement.

The tracking latency is defined as the delay after which the state of the real world is reflected by the output of the tracking system. This delay depends on the following additive components: (1) the sampling interval of the sensors, (2) processing delays in the sensor nodes, (3) the network latency, (4) delays caused by the message ordering algorithm, and (5) delays caused by the algorithm used to compute the target location estimate. The minimum value of (1) heaviliy depends on the sensor and processor

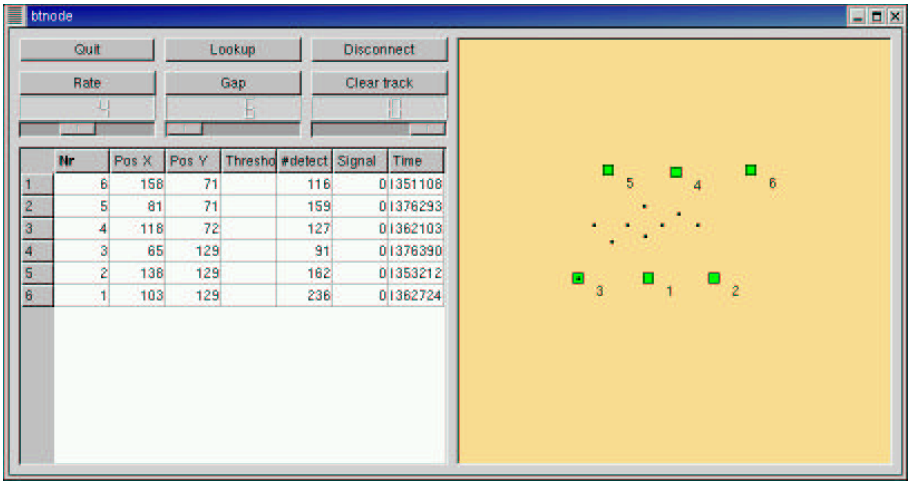


Fig. 7. Measurement setup as shown by the graphical user interface.

hardware. In our implementation, (1) is limited to about 0.1ms by the analog-to-digital converter. Components (2) and (4) are small in our system due to the simplicity of the used algorithms.

To evaluate the tracking latency of our system, we measured the sum of (2), (3), (4), and (5) by calculating the age of each notification after it has been processed by the location estimation algorithm. During the course of our experiment, the average age was 56ms. We also monitored the value of Δ used by the message ordering algorithm. We used an initial guess of $\Delta = 20$ms. At the beginning of the experiment, this value was quickly adapted to 52ms. Recall from Section 3.5 that messages may be dropped by the ordering algorithm if the value used for Δ is lower than the actual maximum network latency. Surprisingly, during our experiments not a single message was dropped. This is due to the fact that the time between arrival of successive notifications at the base station was always greater than the network latency in our experiment. However, this is typically not the case for a real deployment, where the network latency can be significantly larger and where many densely deployed nodes may detect the target almost concurrently and generate according notifications in short intervals.

Note that the above values give only a rough impression of the performance of the tracking system, since we had only 6 sensor nodes available. We plan to build the necessary sensor hardware for a significantly larger number of sensor nodes, which will allow us to perform more thorough measurements.

Figure 7 shows the above measurement setup as depicted by the graphical user interface. In the top left, a number of controls are shown to lookup sensor nodes (“Lookup”), to disconnect from the sensor nodes (“Disconnect”), to adjust the frequency of sensor readout (“Rate”), to control the detection threshold (“Gap”), and to clear the displayed track (“Clear track”). The table below the controls contains one line for each sensor node, showing x and y position, the current detection threshold, number of detections, the currently detected signal strength, and the time of the last detection. On the right, a

display of the tracking area is shown, depicting the sensor nodes (larger rectangles) and some of the location estimates of the car (smaller squares) moving from right to left.

5 Discussion

The use of passive optical communication allows the construction of tiny and energy efficient sensor nodes compared to current radio-based COTS devices. However, it must be emphasized that this mode of communication implies a number of drawbacks. Besides requiring a free line of sight for communication, Smart Dust networks have a single-hop topology, where sensor nodes can only communicate with a base station transceiver. That is, Smart Dust nodes cannot talk to each other directly. Additionally, the base station can become a scalability bottleneck, since all communication must pass through the BST.

In the presented tracking system, we tried to achieve scalability despite these limitations. This is achieved in part by strictly avoiding inter-node communication. Additionally, the algorithms employed in the BST are designed to be independent of the actual number of nodes in the network. Instead, the overhead of the base station algorithms depends on the number of *active* nodes – those that currently "see" the tracked target. Also, the base station only has to store state information for active nodes.

Despite some similarities in the communication scheme (both true Smart Dust and our prototype use a single-hop network topology), there is one important difference between our prototype and a system based on true Smart Dust. While in our system nodes can send messages to the BST at any time, communication with a Smart Dust node requires that the BST points its laser beam at that particular node. Even though the developers envision a slightly defocused laser beam to enable the BST to communicate with many nodes at a time, the laser has to sweep over the deployment area to give each node a chance to talk to the BST. We deal with the resulting long and variable network delays by introducing a message ordering technique which adapts to the actual maximum network latency.

The data fusion algorithm used in our system might seem somewhat simplistic compared to many approaches described in the literature. However, it achieves a reasonable accuracy while only making a minimum of assumptions about the tracked target. The loss of accuracy can be compensated by increasing the node density – which is be possible due to the expected small size and low cost of Smart Dust nodes.

The tracking system has been designed to tolerate node failures, since these are likely to happen. Messages lost due to node failures will only affect the accuracy of the estimated track. However, this can be compensated by a higher node deployment density.

6 Related Work

Current sensor network research is mainly focusing on COTS Dust platforms such as [24], thus developing algorithms, protocols, and systems that typically depend on the characteristics of these platforms, and which are therefore often not portable to true Smart Dust. Systems for fine-grained sensor node localization, for example, often rely

on direct node-to-node communication or use ultrasound [15]. Approaches for clock synchronization often ignore the fact that time synchronization in networks of sensors is often only needed when and where an "event" occurs, thus wasting energy for continuously synchronizing all clocks [5].

Algorithms for localization and tracking of targets using networks of sensors have been extensively studied in the literature [3, 8]. However, there are only few reports on actual tracking systems with complete support for node localization and time synchronization. Many systems like [27], for example, place sensor nodes at well-known positions and cannot deal with node mobility. Systems for target localization like [19] are often too slow to track the location of a moving target in real-time. Moreover, the employed Time-Difference-Of-Arrival (TDOA) method limits this system to targets which emit acoustic or other signals with low propagation speeds. We are not aware of tracking systems that have been specifically designed for true Smart Dust.

7 Conclusion and Outlook

We have presented a complete proof-of-concept system for tracking the location of real-world phenomena with Smart Dust, using a remote-controlled toy car as a sample target. We presented approaches for target location estimation, node localization, time synchronization, and message ordering that match the requirements of Smart Dust. Since target location estimation is solely based on detecting the proximity of the target by individual Smart Dust nodes, the presented tracking system should be applicable to a wide range of possible target types.

As one of the next steps we want to base car detection on the car's acoustic signature, and plan to evaluate the system using a larger number of sensor nodes. The ultimate goal is to reimplement the system using true Smart Dust.

8 Acknowledgements

The work presented in this paper was supported in part by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

References

1. I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless Sensor Networks: A Survey. *Computer Networks*, 38(4):393–422, March 2002.
2. N. Bulusu, J. Heideman, and D. Estrin. GPS-less Low Cost Outdoor Localization for Very Small Devices. *IEEE Personal Communications*, 7(5):28–34, October 2000.
3. J. C. Chen, K. Yao, and R. E. Hudson. Source Localization and Beamforming. *IEEE Signal Processing Magazine*, 19(2):30–39, 2002.
4. L. Doherty, K. S. J. Pister, and L. E. Ghaoui. Convex Position Estimation in Wireless Sensor Networks. In *Infocom 2001*, Anchorage, Alaska, April 2001.

5. J. Elson and K. Römer. Wireless Sensor Networks: A New Regime for Time Synchronization. *ACM SIGCOMM Computer Communication Review (CCR)*, 33(1):149–154, January 2003.
6. B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. *Global Positioning System: Theory and Practice, 4th Edition*. Springer-Verlag, 1997.
7. L. Lamport. Time, Clocks, and the Ordering of Events in a Distributed System. *Communications of the ACM*, 21(4):558–565, July 1978.
8. D. Li, K. D. Wong, Y. H. Hu, and A. M. Sayeed. Detection, Classification, and Tracking of Targets. *IEEE Signal Processing Magazine*, 19(2):17–29, 2002.
9. M. Mansouri-Samani and M. Sloman. GEM – A Generalised Event Monitoring Language for Distributed Systems. *IEE/IOP/BCS Distributed Systems Engineering Journal*, 4(25), February 1997.
10. D. L. Mills. Improved algorithms for synchronizing computer network clocks. In *Conference on Communication Architectures (ACM SIGCOMM'94)*, London, UK, August 1994. ACM.
11. G. J. Nelson. *Context-Aware and Location Systems*. PhD thesis, University of Cambridge, 1998.
12. K. Römer. Time Synchronization in Ad Hoc Networks. In *MobiHoc 2001*, Long Beach, USA, October 2001.
13. K. Römer. The Lighthouse Location System for Smart Dust. In *MobiSys 2003*, San Fransicso, USA, May 2003.
14. C. Savarese, J. M. Rabaey, and K. Langendoen. Robust Positioning Algorithms for Distributed Ad-Hoc Wireless Sensor Networks. In *USENIX Annual Technical Conference*, Monterey, USA, June 2002.
15. A. Savvides, C. C. Han, and M. Srivastava. Dynamic Fine-Grained Localization in Ad-Hoc Networks of Sensors. In *Mobicom 2001*, Rome, Italy, July 2001.
16. Y. Shang, W. Ruml, Y. Zhang, and M. Fromherz. Localization from Mere Connectivity. In *ACM MobiHoc 2003*, Annapolis, USA, June 2003.
17. Y. C. Shim and C. V. Ramamoorthy. Monitoring and Control of Distributed Systems. In *First Intl. Conference of Systems Integration*, pages 672–681, Morristown, USA, 1990.
18. R. Viswanathan and P. Varshney. Distributed Detection with Multiple Sensors: I. Fundamentals. *Proceedings of the IEEE*, 85(1):54–63, 1997.
19. H. Wang, J. Elson, L. Girod, D. Estrin, and K. Yao. Target Classification and Localization in Habit Monitoring. In *IEEE ICASSP 2003*, Hong Kong, China, April 2003.
20. R. Want, A. Hopper, V. Falcao, and J. Gibbons. The Active Badge Location System. *ACM Transactions on Information Systems*, 10(1):91–102, 1992.
21. B. Warneke, M. Last, B. Leibowitz, and K. S. J. Pister. Smart Dust: Communicating with a Cubic-Millimeter Computer. *IEEE Computer Magazine*, 34(1):44–51, January 2001.
22. B. A. Warneke, M. D. Scott, B. S. Leibowitz, L. Zhou, C. L. Bellew, J. A. Chediak, J. M. Kahn, B. E. Boser, and K. S. J. Pister. An Autonomous 16 cubic mm Solar-Powered Node for Distributed Wireless Sensor Networks. In *IEEE Sensors*, Orlando, USA, June 2002.
23. K. Whitehouse and D. Culler. Calibration as Parameter Estimation in Sensor Networks. In *Workshop on Wireless Sensor Networks and Applications (WSNA) 02*, Atlanta, USA, September 2002.
24. Berkeley Motes. www.xbow.com/Products/Wireless_Sensor_Networks.htm.
25. BTnodes. www.inf.ethz.ch/vs/res/proj/smart-its/btnode.html.
26. DCF77 Radio Time Signal. www.dcf77.de.
27. The 29 Palms Experiment: Tracking vehicles with an UAV-delivered sensor network. tinyos.millennium.berkeley.edu/29Palms.htm.