Algorithms for Advanced Clandestine Tracking in Short-Range Ad Hoc Networks

Saif Al-Kuwari[1,2] and Stephen D. Wolthusen[1,3]

[1] Information Security Group, Department of Mathematics, Royal Holloway, University of London, Egham Hill, Egham TW20 0EX, United Kingdom
[2] Information Technology Center, Department of Information and Research, Ministry of Foreign Affairs, P.O. Box 22711, Doha, Qatar
[3] Norwegian Information Security Laboratory, Gjøvik University College, P.O. Box 191, N-2802 Gjøvik, Norway

Summary. Law enforcement tracking applications are usually required to be passive such that the target is not aware of the tracking process. This passivity requirement can severely affect the accuracy of the tracking process especially in cluttered and densely populated areas. However, short range emissions from mobile devices such as phones and accessories can be used to improve the accuracy of these passive tracking applications. In this paper, we adopt an agent-based clandestine tracking approach where a set of dynamically recruited tracking agents observe single or multiple targets and report to single or multiple trackers. We also describe a few supporting mechanisms and algorithms for security and fault-tolerance.

Keywords: Passive tracking, multiple targets, multiple trackers, agents.

1 Introduction

Mobile devices, ranging from media players and mobile phones to accessories, use different radio frequency communication channels, including IEEE 802.11 and Bluetooth. The ubiquity of such devices and their use by the general public makes them attractive tracking tools. While cellular-based tracking is an established tracking approach, it offers a limited temporal and spatial resolution causing severe disadvantages in rapidly evolving situations where adding or switching between targets is desirable. On the other hand, short-range radio frequency emissions are typically used more often allowing for slightly easier (passive) tracking that does not require any infrastructure or interaction with network operators and can hence be set up rapidly on an ad hoc basis.

In this paper we extend a previous work on using ad hoc short-range networks of dynamically recruited *observation agents* [2] to add functionalities for multiple-target tracking. This potentially allows for detecting and tracking associates of targets or multiple devices carried by the same target. While such an individual may frequently change emission devices to obscure tracking, the mere fact that other devices such as other mobile phones, are carried by the

A.U. Schmidt et al. (Eds.): MobiSec 2010, LNICST 47, pp. 67–79, 2010.

same individual for a period of time allows for the formation of hypotheses on the associations of that individual. In this paper, we therefore report on a group of algorithms to track multiple targets by (possibly) multiple trackers while minimizing the observability of coordinating communication and maximizing coverage and tracking accuracy among the dynamically recruited agents.

2 Related Work

Most multi-target tracking algorithms in sensor networks mainly concentrate on energy efficiency. Jiang *et al.* [5] proposed such algorithm utilizing a scheduling scheme to switch the tracking nodes between sleep and awake states. Other algorithms are based on heavy processing filters to improve the tracking accuracy and usually assumes that the trackers are equipped with sufficient resources to cope with the overhead of the tracking computation, e.g. [4]. Recently, distributed tracking became an established approach [7] where the tracking area is divided into sensor cliques, but the tracking scene for such algorithms need to be pre-configured.

Binary proximity sensors are also used to track both single [6] and multiple [9] targets. These sensors produce a 1-bit output to indicate the presence of the target(s) in their vicinity. However, while algorithms based on binary sensors perform well in short-range networks, they don't scale for the long-range ones. Research in the feasibility of tracking multiple targets by binary proximity sensors is still less mature; in particular, currently, the binary proximity sensors can, with high probability, detect the presence of targets, but cannot *accurately* count or identify them.

In this paper we don't impose specific requirements on the tracking nodes, while also not assuming that such nodes are capable of carrying out heavy computations such as those required by filters. We instead tackle the problem of multiple target tracking from a generic algorithmic high level approach.

3 Overview

General scenario setting is similar to that of our previous work [2], that is, there are three types of entities involved in our tracking scenario: *(1) Trackers:* entities that initiated the tracking process. We assume that trackers are law enforcement officers. *(2) Targets:* entities being tracked. We assume that targets are suspects or criminals. *(3) Agents:* casual pedestrians that happen to present at the tracking scene and consequently get involved in the tracking process. Agents can either be masters or slaves, see section 3.1.

Agents are *recruited* to become part of the tracking network; this happens in two steps. First, the recruiter (who is either a tracker or a master) randomly searches its range and checks that the entities found are suitable for recruitment, which depends on the purpose of recruitment. The tracker only recruits the agents of the first piconet, while subsequent recruitments are handled by the masters to maintain the tracking network. If an agent (master or slave) is no

longer needed, it is retired. We assume that the recruiter is able to communicate with the agent and install a tracking software [2].

3.1 Piconets

The tracking network is divided into *piconets*, which are small ad hoc networks consisting of one master and up to 7 (active) slaves. In our algorithms, there are two types of piconets: *(1) Tracking Piconet,* where the target is localized. At least 3 members of this piconet have to be in the target's range. The master of this piconet is called tracking master which has a central role in carrying out the tracking process; and *(2) Connecting Piconet,* which delivers the target's tracking information from the tracking piconet to the tracker(s).

In the tracking network, there is only one tracking piconet per target (see section 6 for exceptions) and as many connecting piconets as necessary; also, there is usually one route from the tracking piconet to the tracker—the exception is when the transmission algorithm is activated, see section 9.

3.2 Localization

The target is localized by the tracking piconet. The tracking master is responsible for maintaining at least three members of the tracking piconet in the target's range at all time; it is also responsible for periodically transmitting the target's location updates to the tracker(s) to be stored in a *tracking table* that is shared among all trackers. A technique called *trilateration* is used to localize the target which involves a 2-step location estimation process. First, the three agents estimate the distances between themselves and the target, d_1, d_2, d_3, and then geometric transformation is applied to estimate the location of the target in reference of these agents. Assuming that the agents are not synchronized and don't (usually) possess specialized hardware, probably the only way to estimate d_1, d_2, d_3 is by measuring the strength of the target's signals as received at the agents. However, and because of environmental factors affecting the radio propagation, as well as the orientation of the receiving agent, the signal strength measurements are not always accurate, which introduces an inherent error margin; see section 11. Detailed discussions about localization in this and other approaches are presented in [1] and [2].

4 Advanced Tracking Scenarios

Tracking can be handled by single or multiple trackers to track single or multiple targets. We assume that both the trackers and the targets are mobile entities, this suggests four possible scenarios: *Scenario 1:* Single Tracker & Single Target, *Scenario 2:* Single Tracker & Multiple Targets, *Scenario 3:* Multiple Trackers & Single Target, *Scenario 4:* Multiple Trackers & Multiple Targets.

Scenario 1 is the most basic in which all other scenario are built upon, and is covered in [2]. In scenario 2, we introduce *Virtual Tracking*, a technique for

tracking more than one target (see section 6). In scenario 3, we introduce *Handover Tracking*, which involves tracking a target by cooperatively handing over the tracking process between multiple trackers as necessary (see section 5). Finally, tracking in scenario 4 is based on all the above techniques and is the most complex. In all scenarios, we assume that tracker(s) have knowledge of the target(s)' address(es) which is passed to all agents along with the address(es) of all tracker(s). We also assume that trackers possess a key which they share with all recruited agents for lightweight authentication, see section 5.1.

5 Tracking with Multiple Trackers

When target(s) are being tracked by multiple trackers, we assume that a private secure medium linking the trackers is provided (e.g. secure federal network), allowing them to periodically synchronize the tracking information. Clearly, it is usually the case that such network exists in law enforcement setting.

In this scenario, we introduce *handover tracking*, where the trackers cooperatively track the target(s). Usually, the tracking network is handled by a single tracker at any give time, even if multiple trackers are concurrently available (scenario 4 above is the exception, see section 6). However, if the tracking network breaks down, either a tracking handover to another tracker occurs or a temporary tracker is elected until a genuine tracker becomes available.

Ideally, the tracker acknowledges receiving every *target location update* sent by the tracking master. If the tracking master doesn't receive such acknowledgement after 3-4 tracking updates, it assumes the communication between the tracker and itself has failed. The tracking master then senses its own range, if one of the genuine trackers is found, it authenticates it (see section 5.1) and starts transmitting the location updates directly to it. Otherwise, the tracking master elects itself as a *temporary tracker*, creates a temporary tracking table, and breaks the whole tracking network to start a new one from scratch acting as a tracker to recruit agents and form new tracking network. Differently in this situation, the tracking network formed by the temporary tracker is set to be in a *temp mode*. In this mode, the temporary tracker, and beside providing a list of all the genuine trackers and targets to the agents it recruits, it also includes a sensing request, which basically requires all recruited agents to periodically sense their ranges for any of the genuine trackers. If a genuine tracker is detected by a particular piconet (either by its master or slaves), the detecting piconet authenticates the genuine tracker and requests the temporary tracking table from the temporary tracker for the genuine tracker. When the genuine tracker receives the temporary tracking table, it merges it with its perviously synchronized copy of the (outdated) tracking table. Finally, the detecting piconet diverts all location updates to the (new) tracker and retires the redundant piconets.

5.1 Secure Handover

During tracking handover, and since addresses of both trackers and agents can easily be spoofed, it is important to enforce an authentication process between

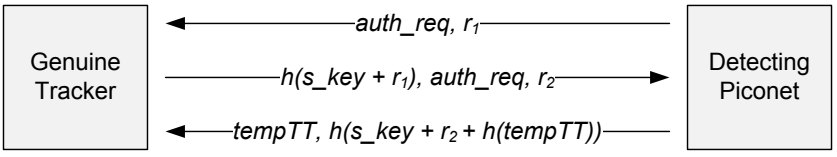

Fig. 1. Mutual Authentication

trackers and agents before handing over the tracking to the trackers. At the time of recruitment, all agents are provided with a key s_key, that is also shared among all trackers, to enforce a 3-way handshake authentication process similar to CHAP (Challenge Handshake Authentication Protocol) [8] while involving random numbers to provide extra layer of protection against replay attacks. Figure 1 depicts how mutual authentication in this scheme takes place.

Once the detecting piconet detects a genuine tracker, the master of that piconet sends an *authentication request* (challenge) to the detected tracker along with a random number r_1. The tracker then adds the random number r_1 to its copy of s_key, hash it $h(s\_key, r_1)$, and sends it back to the detecting piconet along with another authentication request and a random number r_2. Once the response is received, the master of the detecting piconet adds the random number it previously generated (r_1) to its own copy of s_key, hash it and compares the result to the hash it received from the tracker, if they match, the tracker is authenticated and the temporary tracking table (tempTT) is requested from the temporary tracker (similar authentication process can take place between the master of the detecting piconet and the temporary tracker). The tempTT is then hashed $h(tempTT)$, and that hash is further hashed after being added to s_key and the tracker's random number, $h(s\_key, r_2, h(tempTT))$. This final hash is then sent to the tracker with the actual tempTT. Once the tracker receives this response, it tests the integrity of the tempTT (that it hasn't been modified in transient) by calculating $h(s\_key, r_2, h(tempTT))$ and check that it matches the hash it just received. Clearly, this authentication protocol resists man-in-the-middle attack since no adversary can act as a man in the middle (pretending to be an agent for the tracker and a tracker for the agent) unless they possess knowledge of the s_key. Our protocol also resists replay attacks. We consider two replay attack scenarios: (1) when an adversary observing the authentication session captures the authentication information (which is represented by the hash value) to be maliciously used later, and (2) when an agent, either active or retired, is compromised. In scenario (1), an adversary can't use the hash value of a particular session in another since every session adds a random number (which is usually different for different sessions, and is public) to the shared s_key before hashing it; note that it is important to include a random number in this case, otherwise the hash of s_key alone will always be the same (CHAP doesn't include random numbers). In scenario (2), we suggest enforcing a regular change of s_key. Obviously, since both trackers and agents should

have a synchronized copy of the s_key, no s_key change takes place during temporary tracking. In this regard, we assume the existence of a key management mechanism/protocol, but the details of such keying mechanism (key generation, distribution, updating and revocation) are beyond the scope of this paper.

6 Tracking Multiple Targets

When dealing with multiple targets, we introduce *Virtual Tracking*, which entails creating a separate *Virtual Tracking Network* (VTN) for every target. Each VTN consists of virtual connecting and tracking piconets which in turn made up of virtual agents. Every virtual piconet is identified by a Virtual Piconet ID (VPID), and every agent is identified by a Virtual Agent ID (VAID). Physical agents and piconets can have multiple VAID/VPID and be part of multiple VTN's, as shown in figure 2. VPID and VAID format is $xxyy$, such that: in VPID, xx indicates which target the corresponding VTN is tracking, and yy is the piconet ID which is unique within the corresponding VTN; in VAID, xx is the agent ID which is unique within the agent's virtual piconet, and yy is the piconet ID to which the agent belongs, which is unique within the corresponding VTN. This format allows for tracking up to 10 targets; in cases where more than 10 targets may need to be tracked, the xx and yy are expanded as necessary. In multi-tracker multi-target scenario, each VTN is treated as a separate tracking network that may be handled by single or multiple trackers; see algorithm 1.

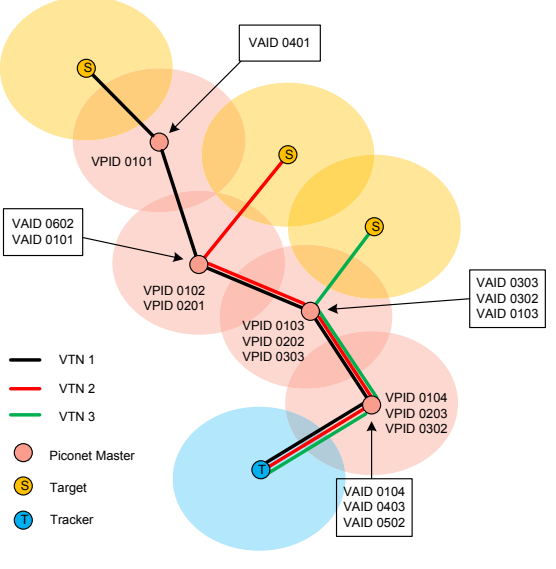

Fig. 2. Virtual Tracking

Algorithm 1. Multi-Trackers Multi-Targets Tracking

```
Tracker[no.ofTrackers]; {List of trackers}
Target[no.ofTargets]; {List of targets}
VTN[no.ofTargets]; {aVTN for every target}
Active_tracker ⇐ Tracker[0]; {choose an initial tracker}
Tracking_table ⇐ Create_tracking_table(Active_tracker);
for i = 0 to i ≤ no.ofTargets do
   VTN[i] ⇐ Create_VTN(targets[i]); {separate VTN for each target}
end for
while Active_tracker = True do
   Update(Tracking_table);
   Sync(Tracking_table); {Sync with other trackers}
end while
{when a gen. tracker becomes unavailable, search for alternative}
for i = 0 to no.ofVTN do
   for j = 0 to no.oftrackers do
      if Search_range(t_master[i], Tracker[j]) = True then
         Active_tracker[i] ⇐ Tracker[j]
      end if
   end for
   Elect_temp(t_master[i]); {if no gen. tracker is found, elect a temp. one}
end for
```

7 Fault Tolerance

In connecting piconets, only two agents are required, but two additional agents are also recruited as backups and paired with the original agents. Backup agents accompany the original agents and both should have the same neighboring agents (unrecruited neighbors don't have to match) so backup agents can seamlessly take over should the agents they are paired with fail. Similar argument applies for the minimum three agents required form the tracking piconets. Every slave/backup-slave pair is monitored by the master of the corresponding piconet by periodically requiring both the slaves and their backup-slaves to provide lists of their neighboring agents to confirm that the two lists for a single pair match; if they don't, the backup slave of that pair is retired and the master of that piconet searches for a replacement. All slaves (and backup slaves) send a periodic *alive* signal to the master, thus any failure is immediately detected by the master. If a slave failed, its corresponding backup slave takes over and a replacement backup slave is recruited. Similarly, the master also has a backup master which should see all the slaves (not necessarily the backup slaves) of that piconet. The master confirms so by periodically requesting its backup master to provide a list of its neighboring agents. The master and the backup master also exchange alive messages. If the backup master didn't receive alive messages from the master for a particular period of time, the backup master forcefully takes over the piconet by sending *mastership* messages to the slaves and can be mutually authenticated as described in section 5.1.

8 Leader Election Algorithm

Occasionally, some agents may become unavailable either by physically moving away or by suddenly going offline (e.g. power loss). It is also possible that trackers depart from the tracking network, or the tracking network may break down at any time and cut off the link connecting the tracker. In all these situations, replacement agents/trackers have to be recruited/elected.

8.1 Tracker Election

Tracker election process is informally described in section 5 and is applicable in both single and multiple target scenarios. This process involves electing a temporary tracker and is activated when the tracking master doesn't receive 3-4 location updates acknowledgements from the tracker; in this case, the tracking master elects itself as a temporary tracker, see section 5 for details, also see algorithm 2 for a pseudocode of the tracker election algorithm.

Algorithm 2. Temporary Tracker Election

```
if Failed_ack = 3 then
  Tracking_master ⇐ Temp_tracker
  Tracking_network ⇐ break
  Search_range(Tracking_master) {Tracking_master senses its range}
  if Gen_tracker = True then
    Divert_traffic(Gen_tracker);
  end if
  Gen_tracker = False;
  Temp_mode = True; {activate temp mode}
  Temp_tracking_table ⇐ Create_tracking_table(Temp_tracker);
  Init_tracking(Temp_tracker); {Temp_tracker initiates the tracking network}
  while Gen_tracker = False do
    Update(Temp_tracking_table);
    Check(Gen_tracker);
    if Gen_tracker = True then
      Gen_tracker ⇐ authenticate;
      Gen_tracker ⇐ Send(temp_tracking_table);
      Divert_traffic(Gen_tracker);
      Gen_tracker = True;
    end if
  end while
end if
```

9 Transmission Algorithm

To improve the passivity of the tracking process, it is important that the transmission of the tracking information doesn't attract the target's attention. Thus,

the tracking information should not always flow over the same route. The transmission algorithm (TA) is comparable to a routing protocol that doesn't *always* use the shortest path between nodes—this may slightly tradeoff efficiency.

The TA is an optional parameter that only trackers can activate. Without the TA, there is only one route between the tracking piconet and the tracker, but once activated, the algorithm is handled by the tracking master and proceeds in two steps: first, the tracking master creates new redundant routes by recruiting additional agents, and then selects which route the target's location updates should take on their way to the tracker. It's important to create such redundant routes only around the target which then can merge into a single route down to the tracker (creating redundant routes beyond this point will not improve the passivity of the tracking because it is not observable by the target either way). Algorithm 3 formalizes these steps, where function $Recruit(x, y)$ recruits any entity that is in the range of both x and y, and function $forward(x, y)$ forwards any traffic received at x to y.

Algorithm 3. Transmission Algorithm

```
if TA = True then {if TA is activated}
  nei ⇐ Get_nei(t_master) {get the neighbors of t_master}
  agent[MAX] {max no. of redundant routes}
  for n = 0 to n ≤ MAX do
    agent[n] ⇐ Recruit(t_master, nei);
    Forward(agent[n], nei);
  end for
  Buffer[MAX-1] {set the buffer size}
  while TA = True do
    n = 0
    while Buffer ≠ Full do
      Send_updates_via(agent[n]);
      Buffer ⇐ Append(agent[n]);
      increment n
    end while
    Purge(Buffer);
  end while
end if
```

9.1 Routes Formation

To form additional (redundant) routes, the tracking master searches its range for agents that are in both its range and the range of its immediate neighboring connecting master. Once a suitable agent is found, the tracking master recruits it and configures it to forward any traffic it receives (i.e. tracking updates) to the neighboring piconet. Every newly recruited agent forms a new route from the tracking piconet to the tracker. Figure 3 shows an example where the tracking piconet forms two redundant routes by recruiting two additional agents.

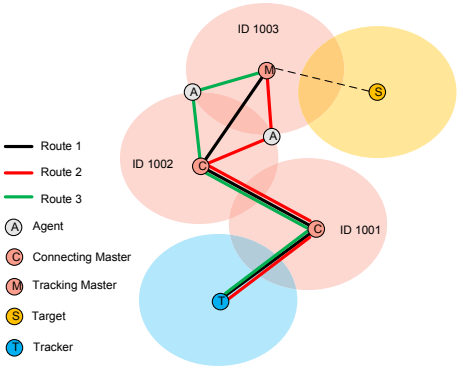

Fig. 3. Transmission Algorithm

9.2 Route Selection

The tracking master selects which route a particular tracking update should take by sending that update through the various redundant agents where beyond these agents, all routes are merged. In fact, and because originally there is only one route between the tracking piconet and the tracker, none of the piconets has to know the exact full route to the tracker. That is, for every connecting piconet, there are only two ports (i.e. agents), a receiving port in which traffic enters the piconet and a sending port in which the traffic exits the piconet—any traffic received by any other port, if any exists, is ignored (in the tracking piconet, there is only one port for sending and receiving). However, when the tracker receives a tracking update, it sends an upstream acknowledgement to the tracking master such that the ports in the intermediate piconets are reversed (the sending port becomes the receiving one and the receiving becomes the sending).

When TA is activated, the tracking master maintains a small buffer to remember the addresses of the last few (redundant) agents it sent the tracking updates through, this is important to distribute the tracking updates transmission over as many routes as possible to prevent over utilizing a particular set of routes. In particular, if there are n routes, the tracking master should buffer approximately $n-1$ routes and only send the next tracking update on a route that is not listed in that buffer, when the buffer is full, any extra record pushes the oldest record out the buffer.

10 Simulation Results and Discussion

In this section we investigate the effect of mobility models and node density on the tracking process. As discussed earlier, if no tracker is available to receive the tracking updates, the tracking network goes through a temporary tracking

period where the tracking master is elected as a temporary tracker until a genuine tracker is detected. However, it is important to investigate how long these *temporary periods* may last because the agents usually have limited resources and may not be able to hold the temporary tracking table for long time. Figures 4, 5 and 6 show how long on average a single temporary period (from losing contact with the trackers until finding them again) lasts in scenarios with different node density (20, 50, 50, 100, 120 nodes) and when adopting different mobility models (Random Waypoint, Brownian Walk, Gauss-Markov mobility models). In these scenarios, the agents have a communication range of $10m$ and moving over an area of $250m^2$ where 10 targets are being tracked by 3 trackers. Furthermore, and to simulate the worst case scenario, we assumed that the agents have limited resources, so we triggered the agents to only form small tracking networks. Note that figures 4, 5 and 6 are illustrated as stacked marked lines to visually show the tenancy of temp time to increase or decrease while increasing the node density.

As the figures show, and regardless of the mobility model, the node density does (even slightly) increase the *temporary tracking* period. However, this is more exemplified in the scenario where Random Waypoint mobility model (figure 4) is adopted, which indicates that the choice of mobility model, and how sophisticated it is, can affect the simulation results.

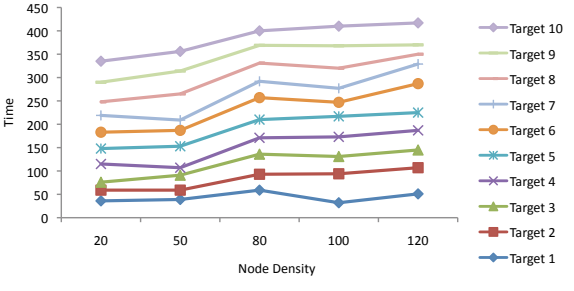

Fig. 4. Results when adopting the Random Waypoint model

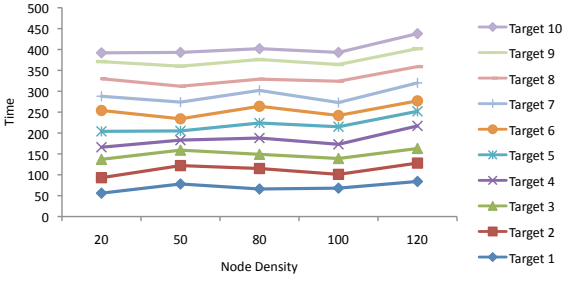

Fig. 5. Results when adopting the Brownian Walk model

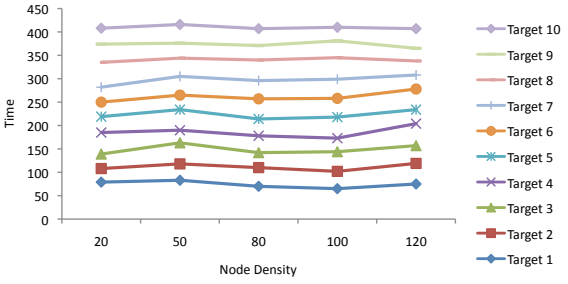

Fig. 6. Results when adopting the Gauss-Markov model

11 Accuracy, Observability and Privacy Implications

We note that all RF (Radio Frequency) measurements in the proposed algorithms are susceptible to errors. Therefore, and unless under a line-of-sight condition, the signal strength may not perfectly correlate with distance which affects the localization accuracy. However, careful investigation of the signaling used in a particular technology may lead to a better estimation; for example, in Bluetooth, recent work [3] showed that measuring the Bluetooth's *Received Power Level* usually yields better location estimation than other Bluetooth parameters measurements. Another solution is to use filters, which are basically estimators to estimate missing or corrupted signal parameters. Using filters can be computationally expensive and may not be suitable for some technologies nevertheless, and thus we don't recommend any leaving this decision for the implementer.

An essential requirement for passive tracking is to minimize the observability of the target, that is, beside not being aware of the tracking process, the target shouldn't be able to detect any abnormal activities (e.g. tracking information transmission). In our tracking system, we treated this issue by proposing the transmission algorithm (section 9), but for better unobservability, we make two further recommendations: (a) localization shouldn't take place very rapidly. Since we are tracking individuals, a tracking interval of 4 or 5 seconds seems reasonable having the limited physical moving pace of humans, (b) the same tracking piconet shouldn't track the same target for long time.

We also note that (clandestine) tracking and the dynamic recruitment of agents raise a number of legal and ethical concerns. However, the mechanisms described here are designated for use by law enforcement and, provided that appropriate procedures are followed, are covered by the *Regulation of Investigatory Powers Act (2000)* in the United Kingdom where such measures are justified.

12 Conclusion

In this paper, we presented a number of algorithms for passive wireless ad hoc based tracking considering a few advanced scenarios. In particular, we introduced *virtual tracking* and *handover tracking*. In virtual tracking, multiple targets are

tracked by creating a virtual tracking network for every target. In handover tracking, multiple trackers track single or multiple targets and manage single or multiple tracking networks by continually switching the tracking process appropriately among them based on the targets' movements. Regardless of the scenarios, if no tracker is available to track particular target(s), a temporary target is elected until a genuine tracker is detected. Future research will look more closely at environment modeling to improve the tracking accuracy.

References

1. Al-Kuwari, S., Wolthusen, S.D.: A Survey of Forensic Localization and Tracking Mechanisms in Short-Range and Cellular Networks. In: 1st International Conference on Digital Forensics & Cyber Crime (ICDF2C), vol. 31, pp. 19–32 (2009)
2. Al-Kuwari, S., Wolthusen, S.D.: Passive Ad-Hoc Localization and Tracking in Short-Range Communication. In: 1st International Conference on Next Generation Wireless Networks (NGWS), Melbourne, Australia (2009)
3. Hossain, A., Soh, W.: A Comprehensive Study of Bluetooth Signal Parameters for Localization. In: 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, pp. 1–5 (2007)
4. Hwang, I., Balakrishnan, H., Roy, K., Tomlin, C.: Multiple-target Tracking and Identity Management in Clutter, with Application to Aircraft Tracking. In: American Control Conference (2004)
5. Jiang, B., Ravindran, B., Cho, H.: Energy Efficient Sleep Scheduling in Sensor Networks for Multiple Target Tracking. In: 4th IEEE International Conference on Distributed Computing in Sensor Systems, Berlin, Heidelberg (2008)
6. Kim, W., Mechitov, K., Choi, J.Y., Ham, S.: On Target Tracking with Binary Proximity Sensors. In: 4th International Symposium on Information Processing in Sensor Networks (2005)
7. Sheng, X., Hu, Y.H., Ramanathan, P.: Distributed particle filter with GMM approximation for multiple targets localization and tracking in wireless sensor network. In: 4th International Symposium on Information Processing in Sensor Networks (2005)
8. Simpson, W.: PPP Challenge Handshake Authentication Protocol (CHAP), RFC 1994 (1996), `http://www.ietf.org/rfc/rfc1994.txt`
9. Singh, J., Madhow, U., Kumar, R., Suri, S., Cagley, R.: Tracking multiple targets using binary proximity sensors. In: 6th International Conference on Information Processing in Sensor Networks (2007)
10. Vyahhi, N., Bakiras, S.: Tracking Moving Objects in Anonymized Trajectories. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) DEXA 2008. LNCS, vol. 5181, pp. 158–171. Springer, Heidelberg (2008)