Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Reducing neighbor discovery latency in docking applications[*][#]

Shuai-zhao JIN[†‡1], Zi-xiao WANG[2], Ya-bo DONG[1], Dong-ming LU[1]
[1]*College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China*
[2]*Department of Computer Science, National University of Singapore, Singapore 119077, Singapore*
[†]E-mail: jinszhao@zju.edu.cn

Received July 2, 2018; Revision accepted Oct. 26, 2018; Crosschecked Aug. 15, 2019

Abstract: Neighbor discovery is important for docking applications, where mobile nodes communicate with static nodes situated at various rendezvous points. Among the existing neighbor discovery protocols, the probabilistic methods perform well in average cases but they have aperiodic, unpredictable, and unbounded discovery latency. Yet, deterministic protocols can provide bounded worst-case discovery latency by sacrificing the average-case performance. In this study, we propose a mobility-assisted slot index synchronization (MASS), which is a new synchronization technique that can improve the average-case performance of deterministic neighbor discovery protocols via slot index synchronization without incurring additional energy consumption. Furthermore, we propose an optimized beacon strategy in MASS to mitigate beaconing collisions, which can lead to discovery failures in situations where multiple neighbors are in the vicinity. We evaluate MASS with theoretical analysis and simulations using real traces from a tourist tracking system deployed at the Mogao Grottoes, which is a famous cultural heritage site in China. We show that MASS can reduce the average discovery latency of state-of-the-art deterministic neighbor discovery protocols by up to two orders of magnitude.

Key words: Neighbor discovery; Docking applications; Slot index synchronization; Mobility-assisted slot index synchronization
https://doi.org/10.1631/FITEE.1800412 **CLC number:** TN929.5

1 Introduction

The availability of cheap wireless sensor nodes has made docking applications feasible for widespread deployment (Dutta and Culler, 2008). In docking applications, there are two sets of sensor nodes, static nodes and mobile nodes. The static nodes are placed at fixed rendezvous points and the mobile nodes are attached to moving objects of interest. Some examples of docking applications include the uploading of cargo transit histories (Malinowski et al., 2007), cattle movement tracking during feeding time (Wark et al., 2007), hiker tracking via trail-side waypoints (Huang et al., 2005), and tourist tracking which tracks tourists at cultural heritage sites (Xia et al., 2008).

At the Mogao Grottoes (Li, 2004), a famous cultural heritage site in China, the high number of tourists presents an important risk factor that affects conservation efforts (Tseng, 2015). The movement of tourists can change the microclimate in the caves, which leads to deterioration of the historical artworks such as paintings and sculptures. Thus,

‡ Corresponding author

* Project supported by the National High-Tech R&D Program of China (No. 2012AA101701), the National Key Technology Support Program of China (Nos. 2013BAK01B00 and 2014BAK16B00), and Grotto Temples Digital Protection of Cultural Relics and Equipment Upgrading of National Cultural Heritage Administration Scientific Research Institutes

# A preliminary version was presented at the 12th International Conference on Mobile Ad Hoc and Sensor Systems, October 19–22, 2015, USA

ORCID: Shuai-zhao JIN, http://orcid.org/0000-0002-5176-5607

it is essential to track the movement of tourists, and studies have been carried out to determine the effects on the microclimate in the caves. The current tracking system that is in place consists of static nodes placed in the caves and mobile nodes carried by tour guides. The static nodes emit beacons periodically, which are picked up by the mobile nodes for each tour group that enters a cave. Not only is detection of a tour group important, but also it is essential to accurately measure the duration of stay within each cave. In addition, power is a constraint for the tracking system because modern infrastructure systems cannot be installed in the caves to supply power to the static nodes.

In the current tracking system, the static nodes are configured at a duty cycle of 0.6% (they wake up and send beacons for 30 ms every 5 s) and the mobile nodes are always active and continuously listening for the beacons. While this approach achieves an average discovery latency of 2.5 s, the mobile nodes must be charged daily, which is inconvenient and is also a major impediment to plans to scale up the system. Our goal is to reduce energy consumption by introducing duty cycling for the mobile nodes, without increasing discovery latency.

Neighbor discovery protocols would allow both the static and mobile nodes not to run while ensuring discovery. Through different designs for the wake-sleep patterns, neighbor discovery protocols can ensure that the nodes will wake up in the same time slot to discover each other. The key insight of this work is that discovery latency can be minimized by synchronizing the active and sleeping slots of the nodes. This efficiency can occur because when the slot indices are synchronized, the nodes will discover each other at the next earliest active slot (the next overlapping active slot).

While this might sound straightforward, it turns out that synchronizing the nodes for a docking application introduces a number of challenges. First, the synchronization must be done in a distributed manner because the static nodes cannot directly communicate with each other. Instead, they can communicate by only relaying information through the mobile nodes; that is to say, the mobiles nodes work as proxy. Second, the synchronization should converge quickly to achieve any gain. Finally, the intrinsic inaccuracies in the sensor clocks will result in clock drift. If the synchronization of the slots drifts by just a small amount, the resulting latency can potentially become a worst case instead of being optimal. In other words, synchronization can significantly degrade performance instead of improving performance if we are not careful.

We propose and evaluate a system called mobility-assisted slot index synchronization (MASS), which improves the average-case performance of existing deterministic neighbor discovery protocols via slot index synchronization. In MASS, the nodes first elect a reference node in a distributed manner and then synchronize their slot indices with that reference node. We exploit the natural visiting patterns of tourists to elect the reference node. For our Mogao Grottoes traces, MASS can synchronize all of the 60 static nodes within 3 h, which is less than half of the 10 h operational period of the site. Yet, even if MASS cannot synchronize all the nodes, we can still reduce the discovery latency for the nodes that are synchronized. To mitigate the impact of clock drift, MASS employs existing techniques to estimate the clock skew between the nodes and to compensate for the clock drift with respect to the elected reference node. The experiments show that our sensor nodes could achieve an error of 0.98 ms and 2.4 ms per hour for 50% and 90% of the time respectively, which is well within one slot interval.

Furthermore, beaconing collisions pose another challenge in reducing discovery latency. Existing neighbor discovery protocols focus mainly on pairwise discovery latency and assume that discovery will occur when the active slots overlap. However, when there are multiple neighbors in the vicinity, a beaconing collision will lead to a discovery failure in which case the discovery latency cannot be predicted by existing theoretical analysis. To mitigate beaconing collisions, we propose an optimized beaconing strategy in MASS for existing neighbor discovery protocols and further design a heterogeneous working pattern called ABPL based on the state-of-the-art Searchlight protocol (Bakht et al., 2012).

We evaluate MASS via simulation using a 31-day trace obtained from the existing Mogao Grottoes tourist tracking system and show that MASS can reduce the discovery latency of four existing deterministic protocols, BlindDate (Wang et al., 2013), Searchlight (Bakht et al., 2012), Disco (Dutta and Culler, 2008), and U-Connect (Kandhalu et al., 2010), by about two orders of magnitude over 75%

of the time. Therefore, with MASS, we can achieve a discovery latency of less than 5 s for 70% of the time, even when the duty cycle of the mobile nodes is reduced to 0.5% to save power. By reducing the duty cycle from 100% to 0.5%, our mobile nodes can last 200 times longer and be charged every six months instead of daily.

To the best of our knowledge, we are the first to propose the use of slot index synchronization to improve neighbor discovery latency in docking applications. We initially addressed the resulting challenges using existing techniques such as clock skew estimation and clock drift compensation, and proposed a synchronization process and priority metric in our previous work (Jin et al., 2015). The current study extends our previous work by validating the earlier simulation results with a real testbed implementation. We also focus on the beaconing collision problem when there are multiple neighbors in the vicinity at any time because it is a common case in docking applications.

2 Related work

2.1 Neighbor discovery protocols

Neighbor discovery protocols are needed in duty-cycled networks to ensure discovery. Most neighbor discovery protocols are slotted protocols, dividing time into identically sized discrete slots. In each slot, sensor nodes either sleep or wake up according to various patterns determined by the protocol. During active slots, the nodes wake up to transmit and listen for beacons from potential neighbors. Existing protocols can be classified by their adoption of either probabilistic or deterministic wake-sleep slot schedules to achieve an overlap. In probabilistic protocols, the sleep-wake schedule is based on a randomized function (McGlynn and Borbash, 2001). Generally, probabilistic protocols can achieve a good average-case performance. Unfortunately, the worst-case performance is unbounded; for example, there is a small chance that two nodes will never discover each other.

On the other hand, deterministic protocols provide a bounded worst-case latency because the sleep-wake schedule is designed to ensure discovery within one sleep-wake period. Prime-based protocols such as Disco (Dutta and Culler, 2008) and U-Connect (Kandhalu et al., 2010) guarantee a bounded discovery latency based on the Chinese remainder theorem (Niven et al., 1980). Quorum-based deterministic protocols (Tseng et al., 2003; Lai et al., 2010) have slots that are grouped into a two-dimensional $m \times m$ array for a period of m^2. Each node picks one row and one column of the entries as its active slots, thus ensuring that any two nodes will have at least two overlapping active slots in each period. The state-of-the-art Searchlight (Bakht et al., 2012) protocol further reduces the overlapping slots to at least one by grouping the slots into an array of size $\lfloor m/2 \rfloor \times m$. The duty cycle can be further reduced by striped probing, where the probe skips every two slots. Extending the duration of the active slots ensures overlap and guarantees detection. BlindDate (Wang et al., 2013) is a recently proposed protocol to improve the performance of Searchlight through the use of two dynamic probe slots traversing toward each other in opposite directions within each period. BlindDate also requires the extension of active slots like Searchlight. Sun et al. (2014b) proposed Hello, a generalized framework for deterministic protocols that can represent existing protocols, such as Quorum, Disco, U-Connect, and Searchlight, using a set of parameters. They proved that Searchlight is the optimal symmetric protocol. Our technique, MASS, complements existing deterministic protocols and can significantly reduce discovery latency through slot index synchronization.

Some collaborative neighbor discovery methods have been proposed to further reduce neighbor discovery latency based on the obtained neighbor information. WiFlock (Purohit et al., 2011) combines neighbor discovery and maintenance using a collaborative beaconing mechanism. Acc (Zhang et al., 2012) accelerates discovery in an on-demand autonomous manner. It also leverages knowledge in the neighbor tables of neighbors to accelerate discovery of indirect neighbors. These collaborative techniques pose one trend where some additional information can be used to accelerate neighbor discovery. Such approaches are largely orthogonal and will benefit from the faster discovery latency that our proposed method offers.

2.2 Clock synchronization

There are two common approaches for clock synchronization in multihop wireless networks (Sun et al., 2014a), reference-based clock synchronization and distributed clock synchronization. In reference-based clock synchronization, the nodes synchronize their clocks with respect to one reference node or more reference nodes. These reference nodes are known as roots in tree-based protocols (Ganeriwal et al., 2003; Su and Akyildiz, 2005), gateways in cluster-based protocols (Elson et al., 2002), and time servers in protocols based on network time synchronization (Ye and Cheng, 2008). The main drawback of reference-based protocols is that they are not robust against the failures of reference nodes. Furthermore, they all assume that a reliable means of communication exists between nodes. This assumption does not hold for our tourist tracking system.

In distributed clock synchronization, all nodes run the same distributed synchronization algorithm, which will cause their local clocks to converge to a common global time value. Some techniques to achieve global synchronization include having each node advance its clock to the fastest clock (Sheu et al., 2004; Zhou and Lai, 2007), or to the average clock value of the local nodes (Li and Rus, 2006; Sommer and Wattenhofer, 2009). Glossy is a recent network flooding architecture that achieves an average time synchronization error below 1 μs (Ferrari et al., 2011). However, it is not suitable for docking applications because the mobile nodes are not always connected. To cope with frequent topology changes and long periods of inter-contact, Choi et al. (2012) developed distributed asynchronous clock synchronization (DCS) for delay tolerant networks. Global clock synchronization has been achieved by asynchronously compensating for clock errors using relative clock information that is exchanged among the nodes in DCS. The drawback of distributed synchronization algorithms is that the convergence is slow (typically requiring hundreds of iterations) (He et al., 2012). Furthermore, such protocols work well only in a closed system, where no new nodes join the network once it starts.

3 Case for slot index synchronization

In this section, we describe the background for our target docking application system, and show that slot index synchronization is a promising technique to improve the performance of deterministic neighbor discovery protocols. We also explain why distributed synchronization algorithms are not feasible for our application.

3.1 Tourist tracking at historical sites

The Mogao Grottoes, also known as the caves of the thousand Buddhas, is a world heritage site located in China. They consist of hundreds of caves cut into the side of a cliff, some of which have Buddhist murals painted over the walls and ceilings. Fig. 1 shows a small section of the Mogao Grottoes. Thousands of tourists visit the caves every day, and the high volume of human traffic changes the microclimate in the caves, which can lead to deterioration of the priceless historical artworks. Fig. 2 shows the microclimate changes in cave 332 over the course of

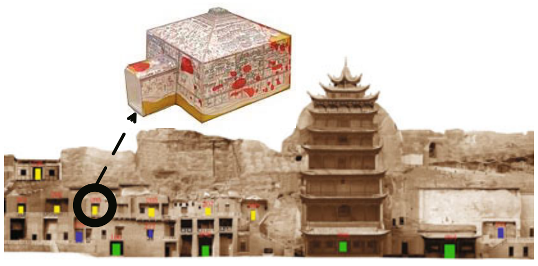


Fig. 1 A small section of the Mogao Grottoes

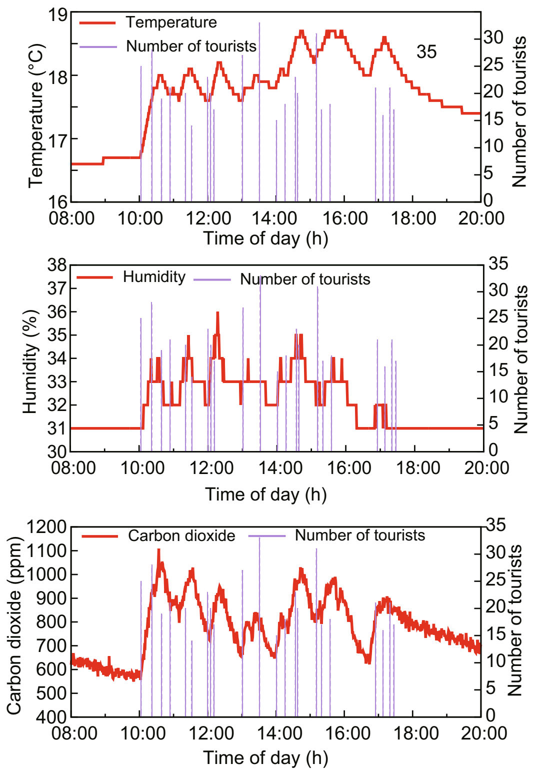


Fig. 2 The number of tourists and microclimate changes in cave 332 in the Mogao Grottoes

a day as tourists are moving in and out. We can see that the temperature, humidity, and carbon dioxide are greatly affected by the number of tourists. Therefore, it is important to monitor and limit the number of visitors to each cave to mitigate the overexposure of the artworks to these hazards due to human contact.

The challenge in tracking tourists at such a historical site is that modern infrastructure such as power supply lines cannot be installed on site. Fixtures are also not allowed to be drilled into the walls for fear of damaging the cultural artefacts. The current tracking system uses static low-power sensor nodes placed at the corners of the caves, which periodically emit signal beacons. Each tour guide holds a mobile sensor node that continuously listens for and logs these beacons as the tour group visits each cave. At the end of the tour, the duration of stay for each tour group in the caves is extracted and recorded from the mobile nodes.

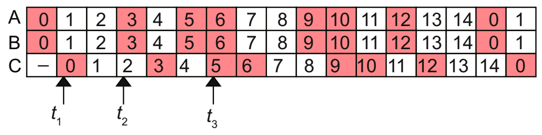
The tourist tracking system at Mogao Grottoes is a typical example of docking applications. Because the cave walls block the radio signals and due to the long distance between the caves, static nodes cannot communicate with each other directly. It is also not feasible to deploy more nodes to create a fully connected wireless sensor network due to the restrictions in place for purpose of conservation, thus limiting the number of nodes being deployed. The lack of power infrastructure also means that static and mobile nodes have to be powered by batteries, which in turn means that energy consumption is a critical consideration.

Neighbor discovery protocols provide a way to allow both the static and mobile nodes not to run, which reduces energy consumption. However, this comes at the cost of increased discovery latency, because the discovery latency and energy consumption are in conflict with each other. The discovery latency is determined by the neighbor discovery protocol, and the duty cycle rate determines the trade-off between discovery latency and energy consumption. In our case, it is equally important to measure the accurate duration of stay for tourists in the caves to monitor the environmental effects of human traffic, which explains the need for small discovery latency.

3.2 Slot index synchronization

Probabilistic neighbor discovery protocols like Birthday (McGlynn and Borbash, 2001) have a lower average-case discovery latency, but the worst-case latency is unbounded. Deterministic protocols like Disco (Dutta and Culler, 2008) and Searchlight (Bakht et al., 2012) have slightly higher average discovery latencies but bounded worst-case latencies. Our key observation is that slot index synchronization has a significant impact on the discovery latency for deterministic protocols.

Fig. 3 shows the discovery process with the parameters $(3, 5)$ between three nodes that are running Disco. This gives a duty cycle of about 50% and the worst-case discovery latency is 3×5 slots. Thus, it has a period of 15 slots. The shaded and white boxes represent active and sleeping slots, respectively. In this example, nodes A and B are strictly synchronized, and node C is out-of-sync by one slot. Supposing that all three nodes come into contact at time t_1, nodes A and B will discover each other at time t_2 with a latency of two slots, while node C would discover the rest at time t_3 with a latency of five slots. It is clear from this illustration that when the nodes are synchronized, the worst-case latency is the largest gap between the active slots. For example, in Fig. 3, the worst-case discovery latency should be three slots instead of 15 slots with slot index synchronization. With slot index synchronization, the active slots of the sensor nodes will always overlap, which means that the discovery latency can be greatly reduced.



A 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 0 1
B 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 0 1
C – 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 0
t_1
t_2
t_3

Fig. 3 Disco with a pair of primes (3, 5)

To investigate the improvement that can be achieved with synchronized nodes, we enumerated all possible slot offsets between a pair of nodes running the same symmetric protocol, and recorded the discovery latency as the number of slots it takes from the first contact to the first intersecting active slot for the two nodes. We treated adjacent active slots as a successful detection as well, because perfect alignment rarely exists in the real world and partially overlapping slots are sufficient for detection. This is especially important for BlindDate and Searchlight-S (Searchlight with striped probing) because perfect alignment may cause a failure of detection. To overcome this, these protocols extend the active slot so as to ensure an overlap of adjacent active slots. We computed the distribution of the discovery latency of the different protocols with duty cycles of 5% and 1% (Bakht et al., 2012; Wang et al., 2013; Sun et al., 2014b), and plotted the result for a 5% duty cycle in Fig. 4. The results for a 1% duty cycle are similar. Our analysis shows that, with the same duty cycle, Searchlight-S performs best with the lowest average- and worst-case latencies.

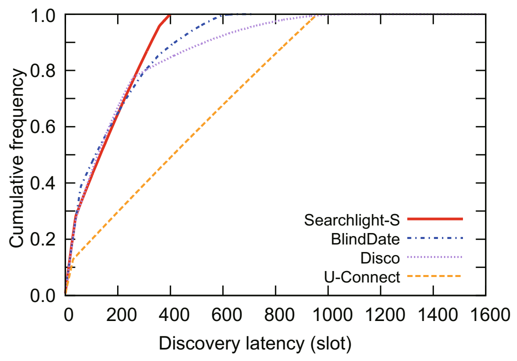


Fig. 4 Cumulative distribution of discovery latency for various protocols at a duty cycle of 5%

We also compared the average- and worst-case latencies of each protocol against the case where both nodes have their slot indices synchronized (Table 1). The key observation here is that the average- and worst-case latencies are significantly lower when the nodes are synchronized on their slot indices. This result is intuitive since two synchronized nodes follow the same sleep-wake pattern and thus wake up at the same time. What is surprising is the amount of improvement. Even for the best performing protocol, Searchlight-S at a 1% duty cycle, the average latency is reduced to 1/72. With the slot size of 5 ms, we can reduce the average latency from 24 s to 0.33 s, and the worst-case latency from 50 s down to about 1 s.

While slot index synchronization improves performance greatly, we cannot always guarantee the synchronization between two nodes due to the inherent errors in the sensor hardware. For example, the timing signals are produced by electronic components, known as oscillators. The oscillation frequency may vary for different reasons such as temperature and vibration, which is known as clock skew. Hence, the effectiveness of slot synchronization is dependent on the accuracy of synchronization.

3.3 Distributed synchronization

In a docking application, static nodes cannot communicate with each other directly and have to rely on mobile nodes to relay information. One approach for achieving synchronization would be to use a distributed clock synchronization algorithm such as DCS (Choi et al., 2012). The duty cycle of the nodes can then be derived from the resulting reference clock, and synchronizing their clocks will naturally synchronize the slot indices.

However, this approach does not work for two reasons. First, we have found that DCS either fails or takes an extremely long time to synchronize the

Table 1 Average- and worst-case discovery latency

Protocol	Overall performance (slot)		Synchronized index (slot)		Average-case latency improvement	Worst-case latency improvement	Parameter(s)
	Average-case latency [s]	Worst-case latency [s]	Average-case latency [s]	Worst-case latency [s]			
			5% duty cycle, 25 ms per slot				
Searchlight-S	151 [3.78]	399 [9.98]	12.3 [0.31]	37 [0.93]	12.28	10.78	40
BlindDate	168 [4.20]	685 [17.13]	13.8 [0.35]	46 [1.15]	12.17	14.89	12
Disco	194 [4.85]	1071 [26.78]	12.7 [0.32]	36 [0.90]	15.28	29.75	(37, 43)
U-Connect	423 [10.6]	960 [24.00]	14.6 [0.37]	30 [0.75]	28.97	32.00	31
			1% duty cycle, 5 ms per slot				
Searchlight-S	4711 [23.56]	9999 [50.00]	65.7 [0.33]	197 [0.99]	71.70	50.76	200
BlindDate	6387 [31.94]	17 821 [89.1]	71.4 [0.36]	238 [1.19]	89.45	74.88	60
Disco	10 125 [50.6]	35 655 [178]	64.1 [0.32]	180 [0.90]	157.96	198.08	(181, 211)
U-Connect	11 123 [55.6]	22 800 [114]	74.6 [0.37]	150 [0.75]	149.10	152.00	151

Values in middle brackets indicate the latency in seconds clocks of the nodes. This means that either node indices will never be synchronized, or the improvement would be small due to the long convergence time. Second, DCS is designed to work in a closed system, where no nodes leave or join the system. Otherwise, it can no longer guarantee convergence. In our docking application, the mobile nodes can leave and new mobile nodes can enter the system at any time.

Because of the shortcomings of the distributed algorithms, we adopt a reference-based technique for synchronization, where one node is elected to be the reference node, with which all other nodes are synchronized. The key observation in our work is that although the mobile nodes do not follow a predetermined path, the movement is not completely random. For this reason, we can exploit the movement patterns of the tourists and design a simple algorithm to elect the reference node.

4 MASS

We have shown that slot index synchronization can significantly reduce neighbor discovery latency under the same duty cycle. In this section, we describe the design of MASS for our tourist tracking application at the Mogao Grottoes.

In the tourist tracking system, all static and mobile nodes will adopt the same neighbor discovery protocol with a common set of parameters. Thus, if all (most) of the static nodes have their slot indices synchronized, mobile nodes can likewise have their slot indices synchronized with every static node. This results in a very small discovery latency when the mobile node comes into contact with any of the synchronized static nodes. In docking applications, mobile nodes will continuously discover different static nodes; thus, the slot index synchronization between static nodes can also help reduce the overall discovery latency.

Since static nodes cannot directly communicate with each other, mobile nodes are used to relay information between the static nodes. The challenge is that mobile nodes do not move along pre-determined paths. Thus, we cannot preliminarily determine which static node will be visited next. However, we observed from the traces of our real-world application that the paths of the mobile nodes tend to follow a certain set of patterns. We can exploit the patterns to elect relatively stable reference nodes in a distributed manner, and have the non-reference nodes synchronize their slot index with these reference nodes. In addition, we use the reference node clock as the reference clock to compensate for the clock drift.

4.1 Distributed reference election and synchronization

To elect the reference nodes, we introduce the notion of a priority metric P. Each static node s computes a priority P_s based on the information carried by the mobile nodes in a distributed way. In addition, each static node s stores the recorded priority of its reference, $P_{s.\mathrm{ref}}$, obtained from the mobile nodes. Similarly, each mobile node m stores the priority (P_m) of the last static node with which it was synchronized. When mobile node m encounters a new static node s, the static node first updates its priority P_s. Then the priorities P_s, $P_{s.\mathrm{ref}}$, and P_m are compared and exchanged.

If P_m is larger than P_s and $P_{s.\mathrm{ref}}$, then the static node will synchronize its clock and slot index according to that of the mobile node, and set its reference $P_{s.\mathrm{ref}}$ to P_m. Indirectly, the static node will have now synchronized with the last static node with which the mobile node was synchronized. On the other hand, if P_m is smaller than P_s or $P_{s.\mathrm{ref}}$, the mobile node will synchronize its slot index with the static node. It will also set its reference P_m to the larger of P_s and $P_{s.\mathrm{ref}}$. Fig. 5 illustrates the steps of this process and Table 2 summarizes the above update rules. With this simple algorithm, the node with the largest priority value will be elected as the reference node, from which other nodes will get a reference. It remains for us to describe how a static node s determines and computes its priority P_s.

For our reference node election algorithm to work well, not only does the priority P_s need to be a metric that can be easily computed in a distributed way, but also the resulting priority for different static nodes should preferably be distinct. In our docking application, we observed that some caves were more popular than others due to either their intrinsic

Table 2 Summary of update rules for reference node election

Condition	Static node	Mobile node
$P_m > \max(P_s, P_{s.\mathrm{ref}})$	$P_{s.\mathrm{ref}} \leftarrow P_m$	No change
$P_m < \max(P_s, P_{s.\mathrm{ref}})$	No change	$P_m \leftarrow \max(P_s, P_{s.\mathrm{ref}})$

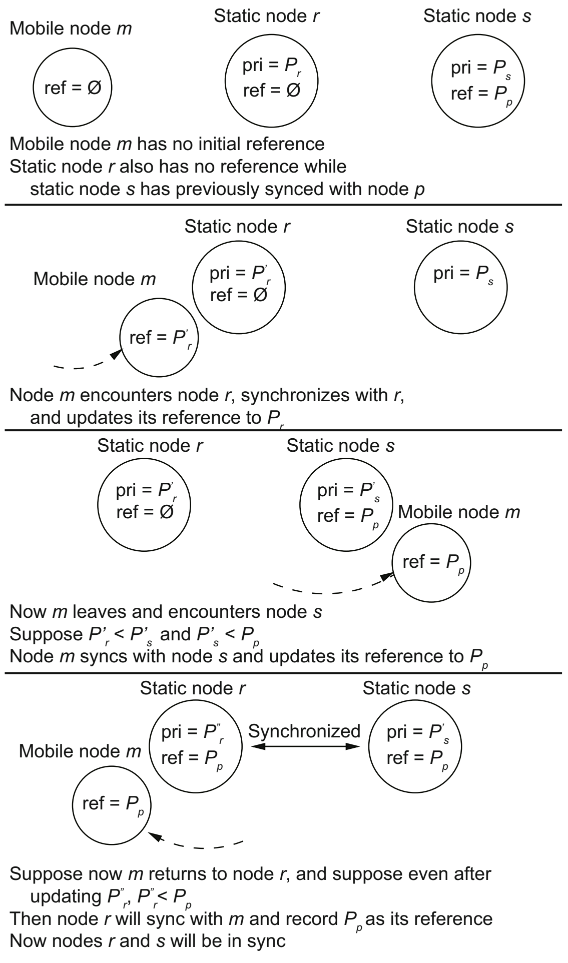

Mobile node *m*
Static node *r*
Static node *s*
ref = Ø
pri = P_r
ref = Ø
pri = P_s
ref = P_p
Mobile node *m* has no initial reference
Static node *r* also has no reference while static node *s* has previously synced with node *p*
Static node *r*
Static node *s*
Mobile node *m*
pri = P'_r
ref = Ø
pri = P_s
ref = P'_r
Node *m* encounters node *r*, synchronizes with *r*, and updates its reference to P_r
Static node *r*
Static node *s*
pri = P'_r
ref = Ø
pri = P'_s
ref = P_p
Mobile node *m*
ref = P_p
Now *m* leaves and encounters node *s*
Suppose $P'_r < P'_s$ and $P'_s < P_p$
Node *m* syncs with node *s* and updates its reference to P_p
Static node *r*
Static node *s*
Mobile node *m*
pri = P''_r
ref = P_p
Synchronized
pri = P'_s
ref = P_p
ref = P_p
Suppose now *m* returns to node *r*, and suppose even after updating P''_r, $P''_r < P_p$
Then node *r* will sync with *m* and record P_p as its reference
Now nodes *r* and *s* will be in sync

Fig. 5 Reference node election

appeal for tourists, or their relative location (it might be near route entrances). Such caves would have more visitors than others. Thus, we decided to use the average inter-arrival time between mobile nodes at each static node s to obtain its priority P_s.

The static nodes compute P by measuring the time elapsed since the last visit to the cave. The priority accumulates using an exponentially weighted moving average (EWMA) with a smoothing factor $\alpha = 1/8$. In other words, the equation to update P_i to P_{i+1} is

$$P_{i+1} = \alpha t + (1 - \alpha)P_i, \tag{1}$$

where t is the elapsed time since the last visit to this cave. Note that in Eq. (1), a smaller value of P means that the priority of the node is higher.

In Fig. 6, we plot the actual average inter-arrival time of each cave over one day of a real trace and the computed priority P_i. Although the computed priority is often not close to the actual average daily values, the trends are similar; that is to say, the more

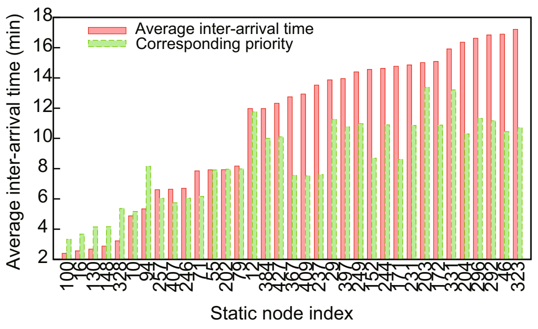


Fig. 6 Average inter-arrival time and priority for different caves at Mogao Grottoes over one day

frequently visited nodes will have a higher priority.

It is entirely plausible that for some docking applications, the static nodes have similar visiting frequencies and the computed priorities might be very close to one another. In such cases, the inter-arrival time might not be the best metric and another metric could be used. However, our reference election algorithm is still applicable if a better metric can be found.

4.2 Clock drift compensation and slot synchronization

Once a reference node is elected, each node will synchronize its clock to that reference node. However, clock drift will introduce errors, even if a node is perfectly synchronized at the start. Hence, both static and mobile nodes in our system must continuously estimate and compensate for clock drift.

4.2.1 Clock skew estimation

There are many existing algorithms for clock skew detection and estimation (Hamilton et al., 2008; Yang et al., 2010; Liao and Barooah, 2013). Zhong et al. (2011) showed that it is possible to continuously detect and estimate the clock skew between two nodes. We believe that any of the state-of-the-art techniques could be used for MASS. Fig. 7 shows one typical technique to estimate the relative clock skew between two nodes, where node A sends

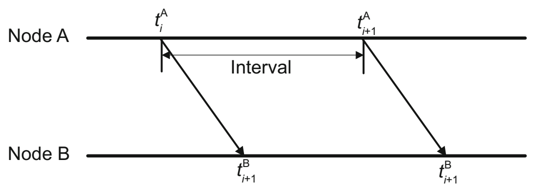


Fig. 7 The clock skew estimation process

beacons periodically and node B keeps listening for the beacons. After receiving two continuous beacons from node A, node B will compute the relative clock skew with node A based on the sending time included in the beacons and the local receiving time, using the following formula (Huang et al., 2013; Xu and Xu, 2013):

$$\delta_{\mathrm{AB}} = \frac{(t_{i+1}^{\mathrm{A}} - t_i^{\mathrm{A}}) - (t_{i+1}^{\mathrm{B}} - t_i^{\mathrm{B}})}{t_{i+1}^{\mathrm{A}} - t_i^{\mathrm{A}}}, \tag{2}$$

where t_i^{A} and t_i^{B} are the local time at the sender's and receiver's clock when the i^{th} beacon was sent and received, respectively. The media access channel (MAC) layer time-stamping technique was used to mitigate the variance of the delivery time (Maróti et al., 2004).

For validation, we performed a simple experiment: A sensor node periodically sent beacons at different intervals to three sensor nodes that kept listening to compute relative clock skew with the sending node. In Fig. 8a, we plot the average estimated relative clock skew and the standard deviation in parts per million (ppm) for beacon intervals from 1 s to 120 s based on the sensor nodes used in the tracking system at the Mogao Grottoes, which will be introduced in Section 5.5. We found that the beacon interval affects the measurement accuracy and a larger beacon interval leads to a more accurate and stable estimation. It is normal since the resolution of the oscillator is not so high and there are some additional delays such as a processing delay on the sensor nodes and propagation delays. If the beacon interval is small, then such jitter will lead to an estimation error. In addition, Schmid et al. (2009) have shown that the larger the beacon interval, the higher the measurement accuracy.

In Fig. 8b, we plot the relative clock skew between the sending node and the three receiving nodes with a beacon interval of 60 s over a few hours. We observed that the three receiving nodes can reliably detect and measure their relative clock skew against the sending node with an error within 1.5 ppm. In Fig. 8c, we plot the time difference between the clocks of two nodes with and without clock drift compensation, which suggests that it is possible to estimate and compensate for the clock drift in our system.

From the real Mogao Grotto traces, tourists (and therefore the mobile nodes) will typically stay in each cave for more than 120 s, which as Fig. 8a suggests, is sufficiently long to obtain a good estimate of the clock skew. Since the average inter-arrival time of the mobile nodes is less than 18 min at each cave (Fig. 6), the expected clock drift during this interval is within one slot (5 ms) with clock drift compensation.

4.2.2 Clock drift compensation

Once we have an estimate of the clock skew, the sensor nodes can compensate for their clock drift against the reference node by adjusting their clocks periodically. Due to the granularity of the clocks (30 μs for our hardware), the compensation interval cannot be too small. We found that having the nodes adjust their clocks every 100 s worked well for our sensor hardware (Fig. 8c), where the clock drift between two nodes could be reduced by up to eight times compared with no compensation.

However, in practice, it is difficult to set an appropriate compensation interval since the relative clock skew between nodes is not known in advance. Thus, instead of using a constant compensation interval, we set a clock drift tolerance to make each node automatically compute the compensation

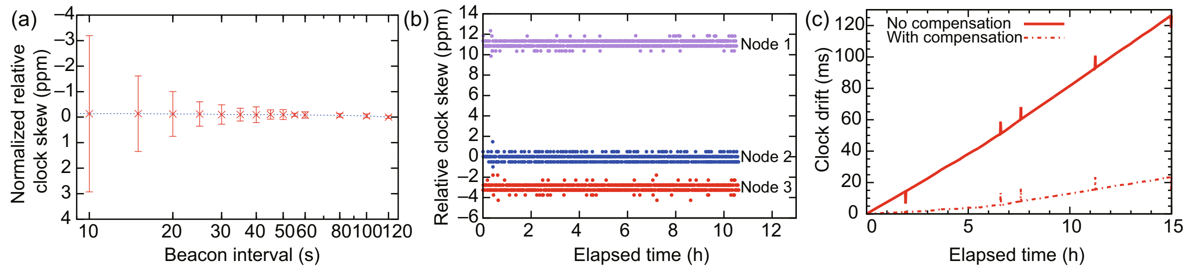


Fig. 8 Clock skew and compensation of sensor nodes: (a) normalized relative clock skew for different beacon intervals; (b) relative clock skew between the sending node and the three receiving nodes; (c) time difference between the clocks of two nodes

interval which we call the dynamic compensation interval. According to the measurements on our hardware, the clock drift tolerance can be set to 1 ms to enable an effective compensation.

We validated this type of dynamic compensation interval using the neighbor discovery protocol. We chose three nodes where one node worked as the static node while the other two worked as the mobile nodes. These three nodes ran the Searchlight protocol at a 5% duty cycle and 5 ms slot size. The mobile nodes performed slot index synchronization with the static node after the first discovery. After 120 s, the two mobile nodes estimated the relative clock skew with the static node. Then each mobile node computed its compensation interval and carried out clock drift compensation periodically.

We computed the wake up time drift after each discovery period and plot the results in Fig. 9. We can see that in the compensation interval, the clock drift grows larger and is revised with compensation. Ideally, with periodic compensation, the mobile nodes can keep slot index synchronization with the static node. However, it is difficult to achieve perfect slot index synchronization in practice due to the measurement errors including the clock skew estimation error, the compensation error, and the dynamic variations in the oscillator frequency. However, with our proposed clock drift compensation method, slot index synchronization can be extended to 5.25 h and 6.15 h compared to the initial 1.15 h and 1.75 h without compensation for the two mobile nodes, respectively.

4.2.3 Slot alignment

While it is possible for the nodes to align their slot timing to a reference node, we found that the resulting improvements from doing so were marginal. As such, we adopted a simple index synchronization process: the nodes simply update the current slot index to the reference node's slot index. Because every node is running the same neighbor discovery protocol, every active slot of such synchronized pairs of nodes will overlap.

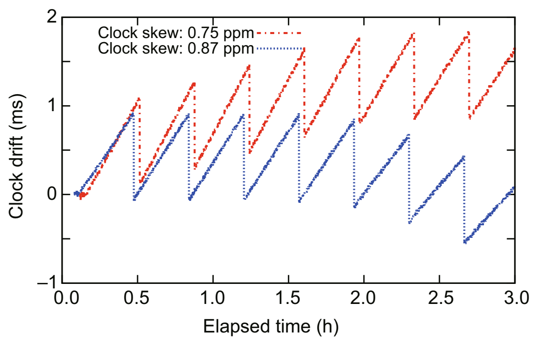


Fig. 9 Clock drift changes with a dynamic compensation interval for two mobile nodes

4.3 Mitigating beaconing collision for multiple neighbors

Most previous work (McGlynn and Borbash, 2001; Tseng et al., 2003; Dutta and Culler, 2008; Kandhalu et al., 2010; Lai et al., 2010; Bakht et al., 2012; Wang et al., 2013; Sun et al., 2014b) focused on neighbor discovery between just two nodes. However, for many applications, it is common for a node to be surrounded by multiple neighbors (Purohit et al., 2011). For instance, in the tourist tracking application at the Mogao Grottoes (Xia et al., 2008), the number of tourists in one cave often exceeds 20 at a time. In such cases, the beaconing collisions between neighbors will become very severe especially when the slot size is small. This trend is alluded in Fig. 10, which plots the probability that at least three nodes will be active (and likely causing collisions) given the presence of overlapping active slots based on the state-of-the-art neighbor discovery protocol Searchlight with the commonly used duty cycles of 1% and 5%.

4.3.1 Impact on discovery latency

The beaconing collisions will cause discovery failures and the discovery latency cannot be predicted using existing theoretical models. To investigate the impact of multiple neighbors on discovery

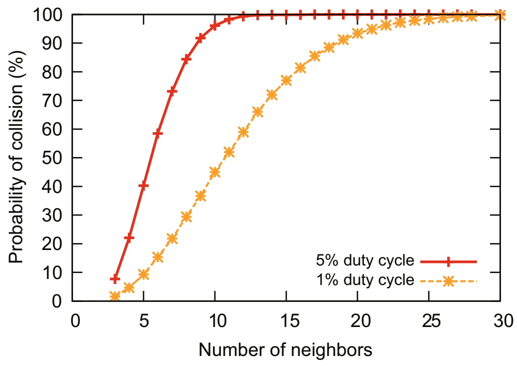


Fig. 10 Probability of at least three nodes being active in an active slot overlap using Searchlight against the number of nodes

latency, we conducted a measurement study based on our hardware platform (introduced in Section 5.5). We chose a pair of nodes and focused on the pairwise latency changes with different numbers of neighbors in the surrounding area. All the nodes ran Searchlight on a 1% duty cycle and slot size of 5 ms. Fig. 11 plots the cumulative distribution functions (CDFs) of the discovery latency between the selected pair of nodes. The results clearly illustrate that collisions caused by having multiple neighbors can cause the latency to balloon to 400+ s in the worst case.

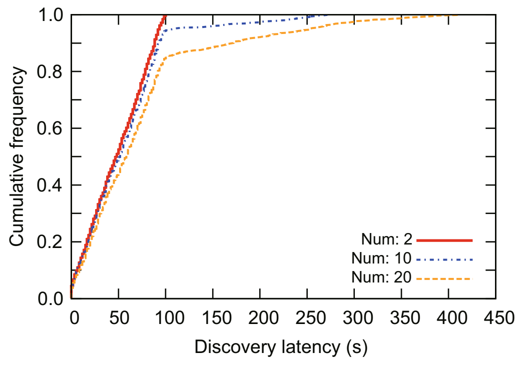


Fig. 11 Cumulative distribution function of the discovery latency between a pair of nodes for the slot size of 5 ms

4.3.2 Reducing the number of beacons

To mitigate beaconing collisions, a natural approach is to reduce the number of beacons in the active slots. Existing deterministic neighbor discovery protocols (Dutta and Culler, 2008; Bakht et al., 2012; Wang et al., 2013; Sun et al., 2014b) generally adopt a beacon-listen-beacon configuration for the active slot. This arrangement seems logical since it enables a pair of nodes to hear each other's beacon and achieve mutual discovery when their active slots overlap. However, since others have shown that sending one beacon is sufficient for mutual discovery (Qiu et al., 2016), we can reduce the number of beacons to just one to mitigate the beaconing collisions in multi-neighbor cases; that is to say, sensor nodes will send only one beacon and then keep listening in each active slot.

Furthermore, we note that for Seachlight the active slot pattern consisted of an anchor slot and a probe slot. We can further reduce the density of beacons by converting the Searchlight probe slot into a listening slot; that is to say, a node will only listen but not beacon during the probe slot. We call such a pattern ABPL and it has been compared to Searchlight in Fig. 12. In this case, one-sided discovery instead of mutual discovery occurs when the active slots overlap. However, we argue that in practice one-sided discovery is sufficient in all (many) applications where mutual discovery can be reconstructed and used offline. Fig. 13 shows the CDFs of pairwise discovery latency with different numbers of neighbors. We can see that reducing the number of beacons can greatly mitigate the beaconing collisions in the multi-neighbor cases. With 20 neighbors in the surrounding area, the beaconing collision rate can be reduced from 15% to 3% and the worst-case discovery latency can also be reduced.

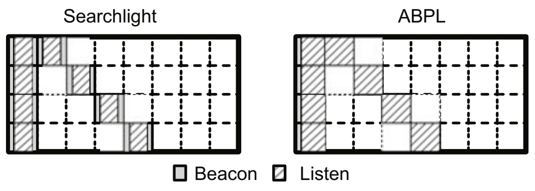


Fig. 12 Comparing Searchlight and ABPL

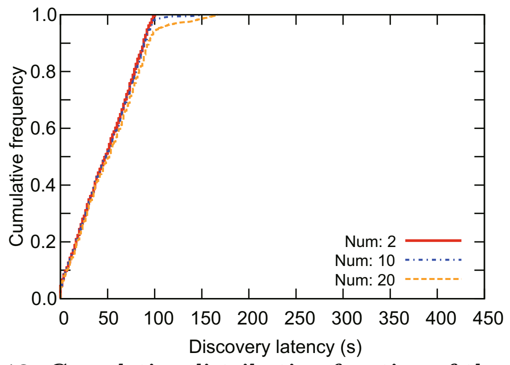


Fig. 13 Cumulative distribution function of the discovery latency between a pair of nodes for the slot size of 5 ms with ABPL

5 Evaluation

In this section, we evaluate the effectiveness of MASS in enhancing existing deterministic neighbor discovery protocols using our custom trace-based simulator as well as real sensor hardware. Our simulator simulates the interactions between the mobile and static nodes using the real traces of tourist movements collected at the Mogao Grottoes. The operating hours of the Mogao Grottoes are from 8:00 am to

6:00 pm daily, so we have approximately 10 h of data for each day. The data set used for our simulations consisted of 31 d of traces over a one-month period in August 2013, containing 8658 movement routes for the mobile nodes and 69 271 cave visits. As an example, the routes for a particular day are shown in Fig. 14, where each cave is assigned a unique identity document (ID).

5.1 Power conservation and latency reduction

In neighbor discovery protocols, the power savings are determined by the duty cycle, which is the proportion of time during which a node stays active. The typical duty cycles used in these protocols were 1% and 5%, and the slot size was typically 25 ms (Bakht et al., 2012; Wang et al., 2013; Sun et al., 2014b).

In Fig. 15a, we plot the cumulative distribution of the discovery latency of the five most common neighbor discovery protocols using a 5% duty cycle and a slot size of 25 ms. We see that as expected, Searchlight-S is the best deterministic protocol, which is consistent with our earlier analysis.

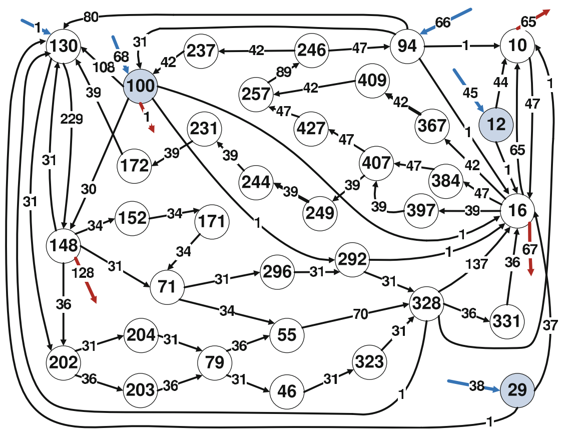


Fig. 14 Visiting route for the tour groups at the Mogao Grottoes on Aug. 1, 2013

The number on the arrows indicates the number of mobile nodes that took that path. Shaded caves indicate the reference nodes elected by the end of the day

Because energy consumption is an important factor in docking applications, we tried to reduce energy consumption by decreasing the duty cycle to 1% (while maintaining the original slot size of 25 ms) and plot the results in Fig. 15b. The discovery latency for all protocols increased by a factor of about 10. Furthermore, the worst-case latency could now be longer than the time that the tourists spent in the caves, which can potentially cause detection failures. We plot the distribution of time that the mobile nodes spent in the caves in Fig. 15c, which confirms our thinking and shows that the worst-case neighbor discovery latency needs to be below 200 s to reliably track tourists.

We can reduce the discovery latency using a smaller slot size. However, how small we can go depends on the hardware and operating system of the nodes. Zhang et al. (2012) suggested that the smallest feasible size for a slot is 5 ms, because any smaller size would cause jitter and make the system unstable. Also, 5 ms of air time could be insufficient for the exchange of usable data. However, we argue that once discovery occurs, the nodes can switch to a different communication protocol for the exchange of information. Thus, we set the slot size to 5 ms with a duty cycle of 1% and plot the new cumulative distribution of the discovery latency in Fig. 16a. We see that as expected, the discovery latency has been improved, but the latency increases to almost 5 times that of the original as in Fig. 15a.

Next, we investigated how synchronization would improve discovery latency. We repeated the

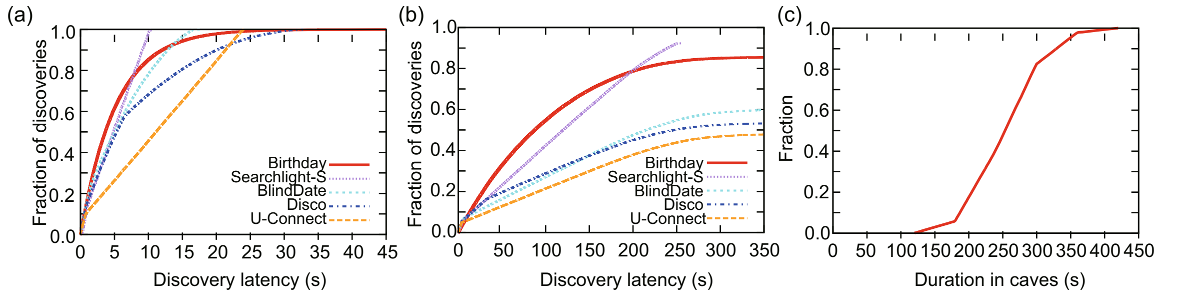


Fig. 15 Impact of reducing duty cycle from 5% to 1% for Birthday, BlindDate, Disco, Searchlight, and U-Connect: (a) 5% duty cycle and 25 ms slot size; (b) 1% duty cycle and 25 ms slot size; (c) length of time that the mobile nodes stay in the caves

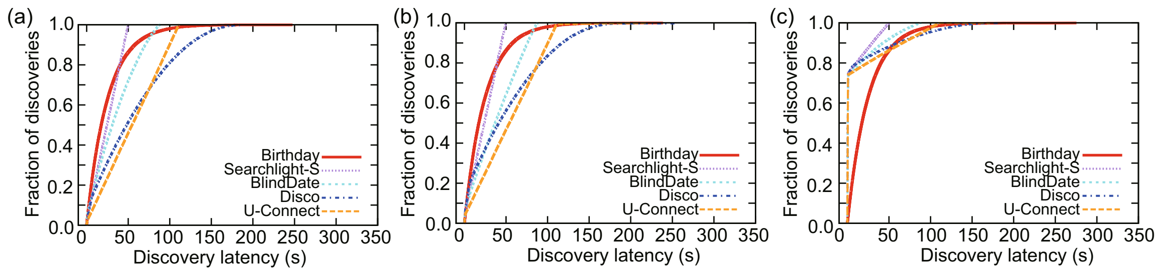


Fig. 16 Discovery latency of each protocol at a 1% duty cycle and 5 ms slot size: (a) no synchronization; (b) distributed asynchronous clock synchronization; (c) mobility-assisted slot index synchronization

simulation with a duty cycle of 1% and a slot size of 5 ms after introducing synchronization with DCS and MASS. We plot the results in Figs. 16b and 16c, respectively.

As expected, synchronization has no impact on the performance of the Birthday protocol. Although in general, synchronization improves the latency for deterministic protocols, the improvements with DCS are marginal. This is because DCS cannot converge fast enough in our scenario. For MASS, the discovery latency for the same protocols was reduced to less than 1 s about 75% of the time, which suggests that most of the nodes were successfully synchronized. However, even with MASS, the worst-case latency remained the same. The reason is that either some nodes could not be synchronized or they had to be re-synchronized at the start of each day. For example, cave 12 and cave 29 at the Mogao Grottoes are usually the first caves to be visited, which means that the mobile nodes cannot relay synchronization information to them. Thus, the static nodes in these two caves cannot synchronize with the selected reference node. If the static nodes cannot be synchronized, the discovery latency between the mobile nodes and static nodes will remain the same as in cases without slot index synchronization; that is to say, the discovery latency still depends on the relative slot offsets.

The reduction in discovery latency is important in our tourist tracking application because our goal is to learn the relationship between the number of tourists and the changes in microclimate such as temperature, humidity, and carbon dioxide in the caves. It is essential to accurately track the duration of stay for tourists in each cave. The median time that visitors spend in each cave is 250 s. Detecting the visitors with a 25 s delay (at 1% duty cycle) results in an error of 10%. By reducing the discovery latency to less than 1 s, we can improve the estimation accuracy by more than 20 times.

5.2 Performance gains using mobility-assisted slot index synchronization

Because the duty cycle affects the discovery latency of neighbor discovery protocols, we plot the percentage of improvement in the average discovery latency with MASS over the original protocols in Fig. 17. An improvement of 90% means that the resulting average discovery latency with MASS is 10% of the original latency. The results show that as the duty cycle increases, the improvement decreases for Searchlight-S and BlindDate. This is because at larger duty cycles, the original protocols already achieved relatively low discovery latencies and there is little room for improvement. On the other hand, Disco and U-Connect show the same improvement across different duty cycles.

Reducing duty cycles is beneficial in saving power, but this comes at a cost in the increased discovery latency. We examined how MASS affects this trade-off by further reducing the duty cycles while

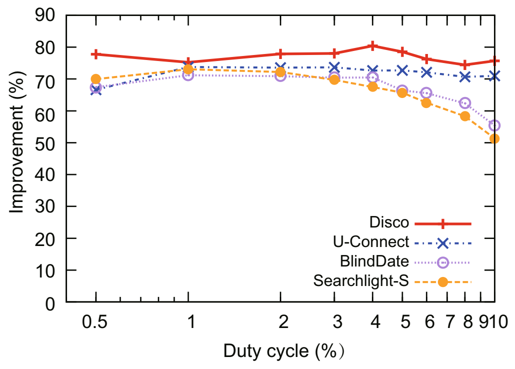


Fig. 17 Improvement for various protocols with mobility-assisted slot index synchronization under different duty cycles with a 5 ms slot size

keeping the slot size at 5 ms. Fig. 18 shows the distribution of the discovery latency for Searchlight-S with MASS using smaller duty cycles. Because Searchlight-S has the lowest worst-case latency, we could reduce the duty cycle to 0.5% while maintaining a 100% discovery success rate in our application. As expected, MASS still manages to keep the discovery latency extremely low within 75% of the time. Decreasing the duty cycle does not affect the improvements of MASS. Only the worst-case latency is affected.

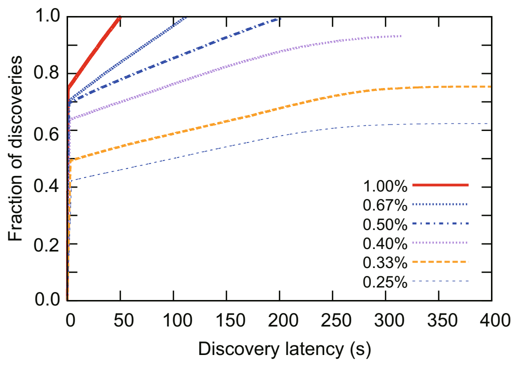


Fig. 18 Distribution of discovery latency of Searchlight-S with mobility-assisted slot index synchronization at very low duty cycles

5.3 Effectiveness of clock skew compensation

Typical clock skew ranges from ±5 to ±100 ppm (Zhong et al., 2011). We showed experimentally that it is possible for clocks to drift apart by 5 ms within just 1 h (Fig. 8c). If the relative clock skew is 20 ppm, and since the average inter-arrival time of the mobile nodes in our application is 15 min (Fig. 6), their clocks would have drifted apart by 18 ms, which is much larger than our slot size of 5 ms. This would suggest that clock drift compensation is essential for ensuring synchronization among nodes.

To validate this line of thinking, we repeated our experiments without clock drift compensation and compared MASS and DCS using the Searchlight-S protocol in Fig. 19. As expected, DCS has no impact on Searchlight-S. Without clock drift compensation, the improvement with MASS is very limited. The same trends are observed for the other neighbor discovery protocols. This demonstrates that clock drift compensation is essential for maintaining synchronicity among nodes.

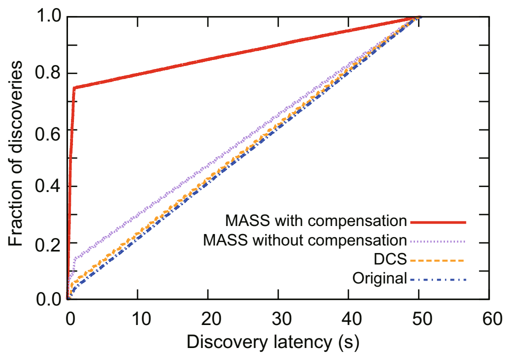


Fig. 19 Distribution of discovery latency for Searchlight-S without clock drift compensation under a 1% duty cycle and 5 ms slot size

5.4 Random traces

Our MASS algorithm exploits the natural visiting patterns of the mobile nodes in a docking application for synchronization. In this subsection, we examine how MASS performs if the visiting patterns are completely random. To ensure a fair comparison, we generated random traces that had the same node distribution as our Mogao Grotto traces.

Fig. 20 shows the distribution of the discovery latencies for Disco, BlindDate, and Searchlight-S with and without MASS. The results show that while MASS could still improve the latency 30% of the time, it is not as effective as before (which was 75% of the time). This is because we used the inter-arrival time as the metric for MASS, which is random in the generated traces. Thus, our reference election algorithm could not maintain a fixed reference node for a prolonged period of time. This highlights that the selection of the metric is a key factor that affects the overall performance for MASS in docking applications.

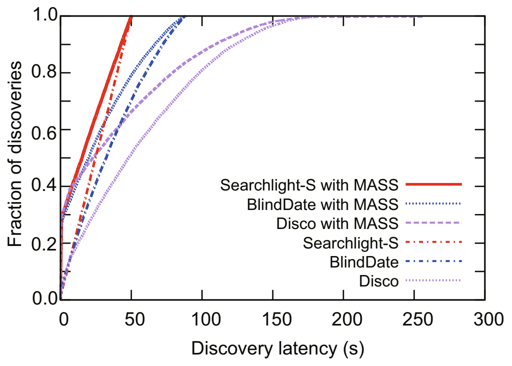


Fig. 20 Distribution of discovery latency with random routes

5.5 Experiments with real sensor hardware

We have shown that the discovery latency of existing neighbor discovery protocols can be greatly improved with MASS. Next, we investigated how MASS performs with real sensor hardware. It is difficult and unrealistic to thoroughly investigate its performance at the Mogao Grottoes since there is already an existing tourist tracking system and we cannot easily replace that system for testing. Instead, we attempted to emulate the setup at the Mogao Grottoes using a mock setup.

We used the same hardware platform in our experiments as the nodes deployed at the Mogao Grottoes (Fig. 21). The sensor node consists of a 32-bit ARM Cortex-M3 MCU SiM3L144 with 64 KB in-system self-programmable flash and a low-power IEEE-compatible radio transceiver AT86RF212 operating at 700/800/900 MHz with a transmission rate from 20 to 250 kb/s. An important feature of the sensor nodes is that they operate at very low power, with a working and sleeping current at 30 mA and 1.2 μA, respectively. A rechargeable 3.7 V Li-polymer battery was used to power the sensor node. The sensor nodes did not run any sensor operating system and were programmed directly in the C language. This is an advantage for our evaluation since it has been shown that sensor operating systems such as TinyOS often introduce additional jitter (Zhang et al., 2012) when the slot size is small.

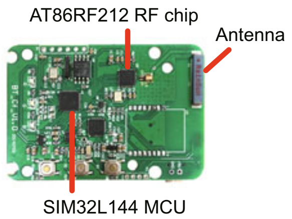


Fig. 21 Custom sensor platform

The mock-up scenario was designed on the third floor of a building (Fig. 22). Ten rooms were selected, and in each room we deployed a static node. To simulate the radio signal blocking phenomenon between the caves at the Mogao Grottoes, the signal transmission power of the static sensor nodes was reduced so that the mobile nodes could hear the beacons only when they entered the room. Then we invited volunteers carrying mobile nodes to move through rooms. To obtain the ground truth for the discovery latency

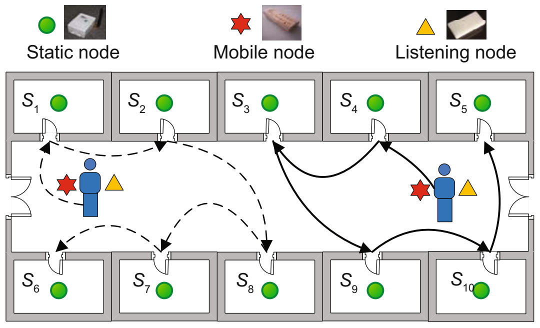


Fig. 22 Mock-up scenario plan in a building

in the mock-up scenario, in other words, the time interval for the mobile nodes coming into the communication range of the static nodes (entering time) to the time they discovered each other, we designated some nodes as listening nodes. The listening nodes worked together with the mobile nodes to record the entering time, which was the time when the node received the first beacon from each static node. Together with the discovered time recorded on the mobile nodes, the corresponding discovery latency could be computed offline.

We adopted the state-of-the-art Searchlight-S with a 1% duty cycle and 5 ms slot size. Each beacon contained a 24-byte payload, which included the necessary information for discovery such as the node ID, slot index, time counters, and priority changes. The total length of the underlying physical message was 41 bytes (4 B of preamble + 1 B of start-of-frame delimiter + 1 B of physical header + 9 B of MAC header + 24 B of MAC payload + 2 B of frame check sequence). With a data range of 250 kb/s, it would take nearly 1.5 ms to transmit one beacon.

To simulate the visiting patterns of tourists at the Mogao Grottoes, we randomly produced the predefined traces for the volunteers based on the statistics from the real traces in one month at the Mogao Grottoes including the visiting sequence and the length of visit distribution in each cave. The experiments lasted three days and there were in total 271 discovery records involving 32 volunteers. Table 3 shows the statistics of the randomly produced moving traces for three days.

5.5.1 Convergence of reference selection

The performance gain with MASS greatly depends on the convergence speed of reference selection (the speed of slot index synchronization). The ideal

Table 3 Randomly produced moving trace

Node	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}
S_1	0	8	2	2	2	2	4	5	2	0
S_2	5	0	3	4	3	3	2	7	3	0
S_3	5	3	0	2	3	5	0	3	2	4
S_4	4	4	4	0	3	1	2	4	4	4
S_5	2	2	2	3	0	4	2	5	2	3
S_6	1	3	6	5	2	0	3	4	4	2
S_7	2	2	1	2	2	3	0	7	4	2
S_8	2	3	2	2	8	5	2	0	2	2
S_9	2	4	4	5	1	4	3	2	0	2
S_{10}	1	2	1	3	2	2	3	5	3	0

For an element at row S_X and column S_Y, the value refers to the number of times that mobile nodes traveled from room S_X to room S_Y. For example, $[S_2, S_3] = 3$ means that the mobile nodes have traveled from room S_2 to room S_3 three times in total in the trace

case is where all the static nodes can maintain slot index synchronization with each other with the mobile nodes moving. However, the convergence speed of reference selection is determined by the moving pattern of mobile nodes and the priority metric adopted. The priority metric may be different in various docking applications.

In this subsection, we examined the convergence speed of MASS based on the priority metric we used in the real traces obtained from the Mogao Grottoes. The average inter-arrival time accumulated using an EWMA with a smoothing factor of $\alpha = 1/8$. Fig. 23 shows the priority for each static node at the end of one day. We can see that the 10 static nodes have different average inter-arrival times, which reflects the moving pattern of the mobile nodes. Note that a smaller value means a higher priority.

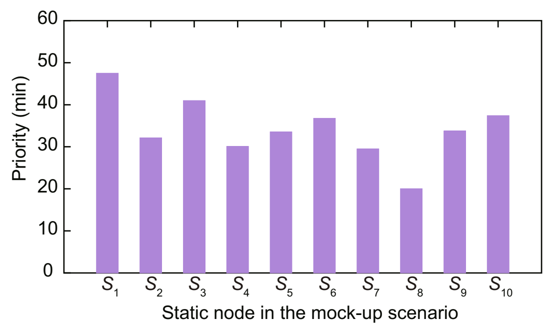


Fig. 23 Priority of the 10 static nodes

Fig. 24 shows the convergence speed of reference selection with the priority metric. We can see that the nine static nodes gradually synchronize with static node S_8 with mobile nodes moving since static node S_8 has the highest priority (smallest average inter-arrival time). It shows that the proposed reference selection algorithm can work well in the mock-up scenario.

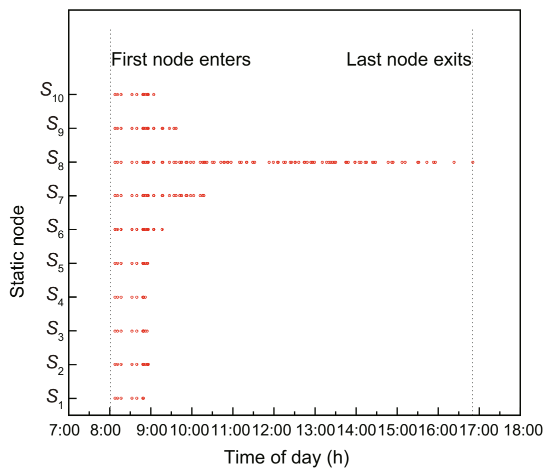


Fig. 24 Reference selection process on the first day

A dot on the graph indicates that the corresponding static node has not finished slot index synchronization with any other static node

5.5.2 Latency reduction

In this subsection, we investigated the reduction in discovery latency that can be achieved with MASS. For comparison, we ran the experiments with and without MASS. First, we asked the volunteers to carry listening nodes and mobile nodes without MASS and then with MASS following the same randomly produced traces. We also ran the simulator with the same traces for comparison.

Fig. 25 shows the cumulative distribution of the discovery latency in the mock-up scenario compared with the simulation results. Two observations can be made from the results. The first is that MASS can reduce the discovery latency in the mock-up scenario. Discovery latency can be reduced to 1 s in 60% of cases. The second observation is that there is a performance gap between simulation results and the results in the practical mock-up scenario. The gap may be caused by the accuracy of clock skew estimation and clock drift compensation, which may break the slot index synchronization. In practice, the clock skew may change according to the environment such as with temperature. The clock drift error accumulates if the synchronization cannot be updated frequently.

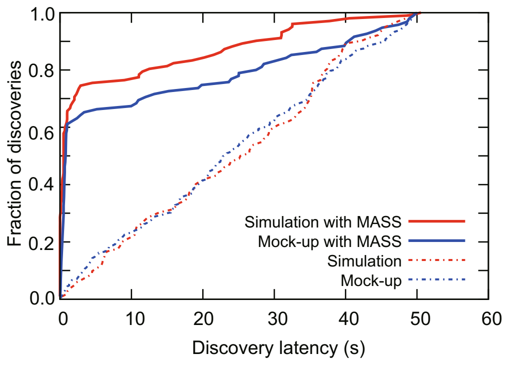


Fig. 25 Discovery latency reduction in the mock-up scenario at a 1% duty cycle and 5 ms slot size

6 Conclusions

In this paper, we showed that slot index synchronization is a practical technique that can significantly improve the average-case performance of deterministic neighbor discovery protocols. By exploiting the mobility patterns of mobile nodes in docking applications, we developed MASS, a reference-based slot index synchronization technique that can improve the average discovery latency, while keeping the energy consumption constant. We showed with simulations based on real traces and experiments with real sensor hardware, that MASS can improve the average discovery latency by up to two orders of magnitude. Our work represents a preliminary investigation into improving existing deterministic neighbor discovery protocols by introducing slot index synchronization. It remains a goal in future work to thoroughly investigate how MASS will perform on real large-scale sensor networks.

Compliance with ethics guidelines

Shuai-zhao JIN, Zi-xiao WANG, Ya-bo DONG, and Dong-ming LU declare that they have no conflict of interest.

References

Bakht M, Trower M, Kravets RH, 2012. Searchlight: won't you be my neighbor? Proc 18th Annual Int Conf on Mobile Computing and Networking, p.185-196. https://doi.org/10.1145/2348543.2348568

Choi BJ, Liang H, Shen XM, et al., 2012. DCS: distributed asynchronous clock synchronization in delay tolerant networks. *IEEE Trans Parall Distrib Syst*, 23(3):491-504. https://doi.org/10.1109/TPDS.2011.179

Dutta P, Culler D, 2008. Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications. Proc 6th ACM Conf on Embedded Network Sensor Systems, p.71-84. https://doi.org/10.1145/1460412.1460420

Elson J, Girod L, Estrin D, 2002. Fine-grained network time synchronization using reference broadcasts. *ACM SIGOPS Oper Syst Rev*, 36(SI):147-163. https://doi.org/10.1145/844128.844143

Ferrari F, Zimmerling M, Thiele L, et al., 2011. Efficient network flooding and time synchronization with glossy. Proc 10th ACM/IEEE Int Conf on Information Processing in Sensor Networks, p.73-84.

Ganeriwal S, Kumar R, Srivastava MB, 2003. Timing-sync protocol for sensor networks. Proc 1st Int Conf on Embedded Networked Sensor Systems, p.138-149. https://doi.org/10.1145/958491.958508

Hamilton BR, Ma X, Zhao Q, et al., 2008. ACES: adaptive clock estimation and synchronization using Kalman filtering. Proc 14th ACM Int Conf on Mobile Computing and Networking, p.152-162. https://doi.org/10.1145/1409944.1409963

He JP, Cheng P, Shi L, et al., 2012. Clock synchronization for random mobile sensor networks. Proc 51st IEEE Conf on Decision and Control, p.2712-2717. https://doi.org/10.1109/CDC.2012.6426053

Huang H, Yun J, Zhong ZG, et al., 2013. PSR: practical synchronous rendezvous in low-duty-cycle wireless networks. Proc IEEE INFOCOM, p.2661-2669. https://doi.org/10.1109/INFCOM.2013.6567074

Huang JH, Amjad S, Mishra S, 2005. CenWits: a sensor-based loosely coupled search and rescue system using witnesses. Proc 3rd Int Conf on Embedded Networked Sensor Systems, p.180-191. https://doi.org/10.1145/1098918.1098938

Jin SZ, Wang ZX, Leong WK, et al., 2015. Improving neighbor discovery with slot index synchronization. IEEE 12th Int Conf on Mobile Ad Hoc and Sensor Systems, p.253-261. https://doi.org/10.1109/MASS.2015.14

Kandhalu A, Lakshmanan K, Rajkumar RR, 2010. U-Connect: a low-latency energy-efficient asynchronous neighbor discovery protocol. Proc 9th ACM/IEEE Int Conf on Information Processing in Sensor Networks, p.350-361. https://doi.org/10.1145/1791212.1791253

Lai SW, Ravindran B, Cho H, 2010. Heterogenous quorum-based wake-up scheduling in wireless sensor networks. *IEEE Trans Comput*, 59(11):1562-1575. https://doi.org/10.1109/TC.2010.20

Li M, 2004. Bodhisattva's crown of the early-Tang period in the Dunhuang art of Mogao Grottoes. *Dunhuang Res*, 2004(6):42-50 (in Chinese). https://doi.org/10.3969/j.issn.1000-4106.2004.06.008

Li Q, Rus D, 2006. Global clock synchronization in sensor networks. *IEEE Trans Comput*, 55(2):214-226. https://doi.org/10.1109/TC.2006.25

Liao CD, Barooah P, 2013. Distributed clock skew and offset estimation from relative measurements in mobile networks with Markovian switching topology. *Automatica*, 49(10):3015-3022. https://doi.org/10.1016/j.automatica.2013.07.015

Malinowski M, Moskwa M, Feldmeier M, et al., 2007. CargoNet: a low-cost micropower sensor node exploiting quasi-passive wakeup for adaptive asychronous monitoring of exceptional events. Proc 5th Int Conf on Embedded Networked Sensor Systems, p.145-159. https://doi.org/10.1145/1322263.1322278

Maróti M, Kusy B, Simon G, et al., 2004. The flooding time synchronization protocol. Proc 2nd Int Conf on Embedded Networked Sensor Systems, p.39-49. https://doi.org/10.1145/1031495.1031501

McGlynn MJ, Borbash SA, 2001. Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks. Proc 2nd ACM Int Symp on Mobile Ad Hoc Networking & Computing, p.137-145. https://doi.org/10.1145/501431.501435

Niven I, Zuckerman HS, Montgomery HL, 1980. An Introduction to the Theory of Numbers (5th Ed.). John Wiley & Sons, New York.

Purohit A, Priyantha B, Liu J, 2011. WiFlock: collaborative group discovery and maintenance in mobile sensor networks. Proc 10th ACM/IEEE Int Conf on Information Processing in Sensor Networks, p.37-48.

Qiu Y, Li SN, Xu XS, et al., 2016. Talk more listen less: energy-efficient neighbor discovery in wireless sensor networks. Proc 35th Annual IEEE Int Conf on Computer Communications, p.1-9. https://doi.org/10.1109/INFOCOM.2016.7524336

Schmid T, Charbiwala Z, Shea R, et al., 2009. Temperature compensated time synchronization. *IEEE Embedd Syst Lett*, 1(2):37-41. https://doi.org/10.1109/LES.2009.2028103

Sheu JP, Chao CM, Sun CW, 2004. A clock synchronization algorithm for multi-hop wireless ad hoc networks. Proc 24th Int Conf on Distributed Computing Systems, p.574-581. https://doi.org/10.1109/icdcs.2004.1281624

Sommer P, Wattenhofer R, 2009. Gradient clock synchronization in wireless sensor networks. Proc Int Conf on Information Processing in Sensor Networks, p.37-48.

Su W, Akyildiz IF, 2005. Time-diffusion synchronization protocol for wireless sensor networks. *IEEE/ACM Trans Netw*, 13(2):384-397. https://doi.org/10.1109/TNET.2004.842228

Sun W, Yang Z, Zhang XL, et al., 2014a. Energy-efficient neighbor discovery in mobile ad hoc and wireless sensor networks: a survey. *IEEE Commun Surv Tutor*, 16(3):1448-1459. https://doi.org/10.1109/SURV.2013.012414.00164

Sun W, Yang Z, Wang KY, et al., 2014b. Hello: a generic flexible protocol for neighbor discovery. Proc IEEE Conf on Computer Communications, p.540-548. https://doi.org/10.1109/INFOCOM.2014.6847978

Tseng SW, 2015. The paradox between conservation and development: can cultural relics of the Dunhuang Mogao Grottoes be preserved through cultural creativity and digital technologies? *Appl Sci Manag Res*, 2(1):39-47.

Tseng YC, Hsu CS, Hsieh TY, 2003. Power-saving protocols for IEEE 802.11-based multi-hop ad hoc networks. *Comput Netw*, 43(3):317-337. https://doi.org/10.1016/S1389-1286(03)00284-6

Wang KY, Mao XF, Liu YH, 2013. BlindDate: a neighbor discovery protocol. Proc 42nd Int Conf on Parallel Processing, p.120-129. https://doi.org/10.1109/ICPP.2013.21

Wark T, Hu W, Sikka P, et al., 2007. A model-based routing protocol for a mobile, delay tolerant network. Proc 5th Int Conf on Embedded Networked Sensor Systems, p.421-422. https://doi.org/10.1145/1322263.1322326

Xia M, Dong YB, Lu DM, et al., 2008. A wireless sensor system for long-term microclimate monitoring in wildland cultural heritage sites. Proc IEEE Int Symp on Parallel and Distributed Processing with Applications, p.207-214. https://doi.org/10.1109/ISPA.2008.75

Xu M, Xu WY, 2013. TACO: temperature-aware compensation for time synchronization in wireless sensor networks. Proc IEEE 10th Int Conf on Mobile Ad-Hoc and Sensor Systems, p.122-130. https://doi.org/10.1109/MASS.2013.52

Yang Z, Pan J, Cai L, 2010. Adaptive clock skew estimation with interactive multi-model Kalman filters for sensor networks. Proc IEEE Int Conf on Communications, p.1-5. https://doi.org/10.1109/ICC.2010.5502549

Ye Q, Cheng L, 2008. DTP: double-pairwise time protocol for disruption tolerant networks. Proc 28th Int Conf on Distributed Computing Systems, p.345-352. https://doi.org/10.1109/ICDCS.2008.73

Zhang DS, He T, Liu YH, et al., 2012. Acc: generic on-demand accelerations for neighbor discovery in mobile applications. Proc 10th ACM Conf on Embedded Network Sensor Systems, p.169-182. https://doi.org/10.1145/2426656.2426674

Zhong ZG, Chen PP, He T, 2011. On-demand time synchronization with predictable accuracy. Proc IEEE INFOCOM, p.2480-2488. https://doi.org/10.1109/INFCOM.2011.5935071

Zhou D, Lai TH, 2007. An accurate and scalable clock synchronization protocol for IEEE 802.11-based multi-hop ad hoc networks. *IEEE Trans Parall Distrib Syst*, 18(12):1797-1808. https://doi.org/10.1109/TPDS.2007.1116