

# Autonomous Proximity Awareness of Bluetooth Devices\*

Changlei Liu  
The Pennsylvania State University  
Dept. of Computer Science and Engineering  
University Park, PA, 16802  
chaliu@cse.psu.edu

Kwan L. Yeung  
The University of Hong Kong  
Dept. of Electrical and Electronic Engineering  
Hong Kong, PRC  
kyeung@eee.hku.hk

**Abstract** – This paper focuses on designing autonomous device discovery algorithms for Bluetooth networks. We first extend the conventional asymmetric Bluetooth link model to three point-to-point symmetric link models. Their performances are compared analytically. To achieve proximity awareness among a group of Bluetooth devices, three control information exchanging methods are also proposed. Combining with the three link models, this gives 9 possible variants of autonomous device discovery algorithm. A comprehensive comparative study based on these 9 variants is then carried out using Bluhoc simulator.

## I. Introduction

Bluetooth [1] is based on frequency hopping spread spectrum technique, where each channel is represented by a pseudo random hopping sequence. Bluetooth devices can be connected either into a star topology, called *piconet*, or form a *scatternet* by linking different piconets together. To construct a scatternet, many scatternet formation algorithms [3-8] have been proposed. Most of them just assume the proximity awareness (i.e. the identities of devices within the transmission range) at each node is known in advance. The mechanism to obtain the proximity awareness, through the process of *device discovery*, is not sufficiently treated.

Device discovery is trivial for single channel network, like 802.11, as each device can directly hear each other via the single broadcast channel. For the multi-channel Bluetooth system, the issue becomes far more complex since two devices within each other's transmission range cannot hear each other unless the link between them is *explicitly* established. Obtaining the proximity awareness in Bluetooth thus constitutes a critical part of the network formation delay (i.e. the time required to establish a connected Bluetooth network).

We can group the earlier work on Bluetooth device discovery into three types, specification-compliant [6], non-specification-compliant [5,7], and those require extra add-on components [9]. For specification-compliant schemes, they are fully compatible with the current Bluetooth specifications and can be readily implemented. For non-specification-compliant schemes, they either need to extend the standard Bluetooth ID packet format [5], or shorten the *de facto* backoff timer [7]. For the proposals relying on extra add-on components [9], such as infrared, RFID, or GPS, the aim is to bypass the sluggish inquiry phase in the Bluetooth link establishment procedure.

In this paper, we focus on designing practical device discovery methods that are compliant to the current Bluetooth specifications. In the next section, we describe the asymmetric link model provided by Bluetooth Specifications. In Section III, the three improved symmetric link models are introduced and

analyzed. Based on them, three new device discovery methods are proposed in Section IV. Performance evaluation is done in Section V and conclusion is presented in Section VI.

## II Asymmetric Link Model

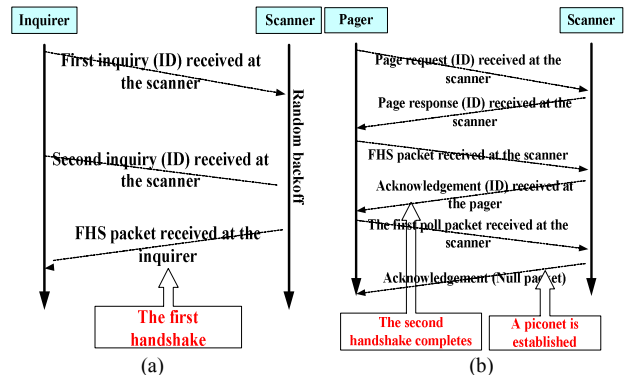


Fig. 1: timing sequence of (a) inquiry handshake and (b) page handshake [1]

Point-to-point Bluetooth link establishment proceeds in two phases, inquiry/inquiry-scan (Fig.1a) and page/page-scan (Fig.1b). In both phases, handshake completes once one device correctly receives the FHS packet from the other, which contains the identity and clock information of the sender. In the inquiry/inquiry-scan phase, the *inquiry handshake* allows the scanner (device) to release its existence to the inquirer (via FHS packet) but not vice versa. The reverse process, concerning how the scanner gets to know the pager (i.e. the inquirer in the previous phase), is done by the *page handshake* in the page/page-scan phase (also via FHS packet). From Fig. 1, it can be seen that the link model provided by Bluetooth is *asymmetric*, i.e. two devices have to be explicitly placed in opposite states (i.e., inquiry vs. inquiry-scan) by users before link establishment. For *autonomous* scatternet formation, we must eliminate this kind of user intervention. In other words, symmetric link models must be designed to allow each device automatically switch between the two states. For more details about how signaling is exchanged in Fig. 1, please refer to [1,4].

## III. Improved Symmetric Link Models

### Model 1: randomized state residence time with fixed state transition

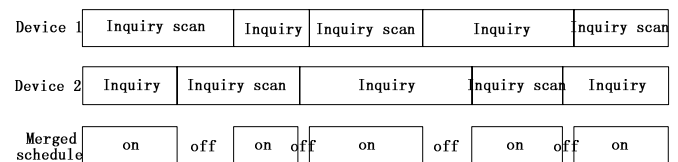


Fig. 2: symmetric link Model 1

\* This work was supported by Competitive Earmarked Research Grant HKU 1180/00E.

The first symmetric link model (Model 1) was adopted in [3,4] for autonomous point-to-point link establishment. As shown in Fig. 2, devices are scheduled to *alternate* between inquiry and inquiry-scan states. The merged schedule follows an on-off process, with an “on-interval” corresponding to the two devices in opposite states, and an “off-interval” otherwise. If the on-interval is long enough, the two devices can complete their link establishment handshakes (Fig. 1). Let the residence time of either on-interval or off-interval be uniformly distributed, the mean *link formation delay* was derived in [3] as:

$$\bar{T} = \frac{b}{8} + \left\{ \frac{b}{4} + \frac{3R}{8} - \frac{b\sqrt{3}}{2R} \arctan\left[\frac{(2R-3b)\sqrt{3}}{3b}\right] - \frac{\sqrt{3}\pi b^2}{6R} + \frac{b}{4} \right\} \left( \frac{4R-b}{b} \right) + \frac{R}{2} \dots (1)$$

*b*: upper bound of the uniform distribution. *R*=640ms: upper bound of the random backoff timer after scanner receives the first inquiry packet.

**Model 2: fixed state residence time with randomized state transition**

Another symmetric link model (Model 2), characterized by fixed state residence time and randomized state transition order, was proposed in [8]. Devices under this model switch states periodically, but pick up their states (i.e. inquiry or inquiry-scan) randomly. The link formation delay was studied via simulations. In [10], we showed that the merged schedule in Model 2 can be extrapolated by a special Bernoulli process. The closed form mean link formation delay can then be obtained as follows. (For ease of numerical computation, a recursive form was also given in [10].)

*p*: fixed state transition probability,  $0 < p < 1$ . *C*: constant state residence time. *R*=640ms.

When  $C > R$ ,  $\bar{T} = \frac{2p(1-p)}{1-2p(1-p)}C + \frac{R}{2}$

When  $C < R$ , assume  $K \leq \frac{R}{C} \leq K+1$

$$\bar{T} = \frac{C}{2} + \frac{\left( \sum_{n=1}^{K-1} \frac{P^n nCR}{R-nC} + C \right) \left( 1 - \sum_{n=1}^{K-1} \frac{P^n nC}{R} - \sum_{n=K}^{\infty} P^n \right)}{\sum_{n=1}^{K-1} \frac{P^n nC}{R} + \sum_{n=K}^{\infty} P^n} + \frac{R}{2} \dots (2)$$

**Model 3: randomized state residence time and state transition**

Combining Models 1 & 2 leads naturally to Model 3, where both state residence time and state transition order are randomized. Assume the state residence time is exponentially distributed, the mean link formation delay can be found [10]:

$$\bar{T} = \frac{1}{4p} \left\{ \lambda + \left[ \frac{\lambda di \log(e^{\frac{2Rp}{\lambda}})}{Rp} + 2(Rp+2) \right] \times \frac{2Rp + \lambda e^{-\frac{2Rp}{\lambda}} - \lambda}{\lambda - \lambda e^{-\frac{2Rp}{\lambda}}} \right\} + \frac{R}{2} \dots (3)$$

*p*, *R* has the same meaning as above;  $\lambda$  is the parameter of exponential distribution.

Fig. 3 compares the performance of the three link models for point-to-point link establishment based on equations (1)-(3). The same mean state residence time is used for all the three models. We can see that Models 1 & 3 have relatively stable performance, while Model 2 is more sensitive to parameter setting; its link formation delay can be as small as 800ms but deteriorates linearly as the mean state residence time grows.

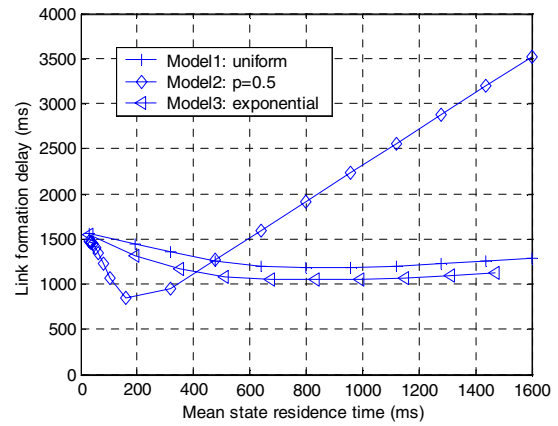


Fig. 3: comparison of the three point-to-point symmetric link models

**Implementation concern**

Although Bluetooth Specifications only depict an asymmetric link model, the Periodic\_Inquiry\_Mode HCI command [1] can be exploited to allow devices to switch between inquiry and inquiry-scan states [4]. Besides, Bluetooth has a pre-determined set of 32-frequency hopping sequences for carrying out the handshaking process shown in Fig. 1. Those hopping sequences are equally split into two subsets, called *A Train* and *B Train*. In order to “hit” the frequency listened to by the scanner, the inquirer/pager has to alternate between A Train and B Train, as it may not know exactly which train the scanner’s frequency falls into. A train repetition strategy is called “*N*-repetition” if a single train has to repeat *N* times before a new train is used. To support symmetric link model, the default 256-repetition strategy should be replaced by 1-repetition for inquiry/inquiry-scan. This is because in Specification 1.1 each inquirer has to repeat 256 times of the same train (A or B) before switching to the other train (B or A) with a minimum of three such switches, the state residence time would be at least  $10ms \times 256 \times 4 = 10.24s$ . This delay is unbearable to the user. We observe that in Specification 1.2 [1], the train repetition strategy for the inquirer is no longer restricted to 256-repetition. This renders more design space to support symmetric link models.

**IV Achieving Mutual Proximity Awareness**

The symmetric link models introduced in the previous section aim at autonomous *point-to-point* link establishment without user intervention. To achieve proximity awareness among a *group* of devices, a specific device discovery method is needed to run on top of such a model. In particular, the following new issues must be addressed. First, each time two devices meet, either via inquiry handshake or page handshake, only unidirectional identification is available. Thus each pair of devices must meet *twice* in order to achieve *mutual* proximity awareness. Second, we may suffer from repetitive/redundant handshakes. This is because the node, that has established a link with one of its neighbors, will not exit the discovery mode immediately, but continue to search until it believes all the proximity information has been collected, i.e. a pre-determined device discovery timer expires.

In the following, three new device discovery methods are proposed for getting proximity awareness among a group of devices.

### Method 1: relying on inquiry handshake only

In Method 1, the two handshakes necessary for mutual information exchange between any two devices occur as inquiry handshake. In particular, when the inquirer receives a FHS packet from the scanner (Fig. 1a) at inquiry/inquiry-scan, the two devices do not proceed to page/page-scan (Fig. 1b) but go back to inquiry/inquiry-scan again. Devices will *never* enter page stage and thus totally avoid duplicate page handshakes. For example, when two devices A & B want to find each other, device A first performs as an inquirer, finds B through the inquiry handshake. Later, when they meet each other again with the reversed roles, B can then discover A.

### Method 2: paging every device discovered

Method 2 follows the usual operation of performing page handshake immediately after the inquiry handshake. Though repetitive/redundant meets at the phase of page/page-scan are unavoidable, the overhead is expected to be small as the page handshake can be quickly finished with the aid of the FHS packet previously collected. Note that once page handshake completes, the ID and clock information of both peer devices are available to each other. Thus it is not necessary to proceed to set up a temporary piconet via further signaling exchange as in [6,7].

### Method 3: paging newly discovered devices only

Method 3 tries to combine the advantages of Method 1 and Method 2. Unlike Method 2, each device maintains a record of devices from/to which the FHS packets have been received/sent. Based on the record, a device can decide if the subsequent page handshake is needed. Now the questions are how to keep an accurate record of FHS packets and how to synchronize the operation of inquirer and scanner. For example, if one device decides to page while the other does not enter page-scan (due to, say, an incorrect record), the paging effort will be wasted.

Without loss of generality, we consider two devices  $i$  and  $j$ . When a FHS packet arrives at  $j$ ,  $j$  records device  $i$  into its array of FHS\_From[ $i$ ]. But how does device  $i$  know that its FHS packet was correctly received by  $j$ ? Due to the possible packet collision, a FHS packet successfully sent does not necessarily mean properly received. We handle this problem by recording destination device  $j$  into FHS\_To[ $j$ ] of device  $i$  only after  $i$  receives the FHS acknowledgement. In case of inquiry handshake, where no FHS acknowledgement will be generated, the FHS packet received from the pager in the subsequent page handshake can be *effectively* treated as an acknowledgement for the FHS packet sent earlier.

The FHS packet record can help to prevent unnecessary handshakes. When the first inquiry handshake completes, the inquirer (device  $i$ ) checks its FHS\_To[ $j$ ] and the inquiry scanner (device  $j$ ) checks its FHS\_From[ $i$ ]. If they have the record of each other, the subsequent page handshake is not needed. But according to the current specification, the ID packet received by the scanner does not identify the sender, and thus scanner  $j$  cannot check FHS\_From[ $i$ ] for the inquirer. In [7], Stefano Basagni *et al.* proposed to extend the ID packet format to incorporate the sender identity. If this can be done, synchronization of the records at the sender and receiver can be easily achieved; otherwise, after the inquiry handshake, the scanner has to enter the page-scan alone until timeout. The timeout value should be large enough for the scanner to detect the pager's existence/absence (based on whether an ID packet is received from the pager in Fig. 1b), but as small as possible to minimize the timeout overhead. By appropriately setting this

timer (PageScanTo in Table I), our simulations in Section V showed that the overhead is justified by the performance gain achieved without extending the ID packet format. Compared with Method 2 where both devices enter page/page-scan each time they meet, Method 3 halves the idle waiting time by keeping only one device in page-scan. In addition, as PageScanTO is shorter than the average time to complete a page handshake, Method 3 essentially introduces less overhead than Method 2.

Note that the values of FHS\_To[ $j$ ] at device  $i$  and FHS\_From[ $i$ ] at device  $j$  are synchronized most of the time, except when the FHS packet is correctly received but its acknowledgement is lost. The out-of-synchronization should be a rare case, which could be amended when the two devices encounter again with reverse roles. To further mitigate the effect of out-of-synchronization, we design the corresponding values for timers PageTo & PageScanTo shown in Table I, and the simulations in Section V are based on those designed values.

### A new timeout approach

Current literature employs a fixed timer to terminate device discovery [6-8]. Its value denotes the whole device discovery period. This approach requires a large timeout value (10.24s by specification) in order to find "all" the neighboring devices. One inherent disadvantage is that fixed timeout values cannot accommodate all kinds of geographical configurations. Therefore, we propose a *relative* timeout approach whose timer refers to the interval between two subsequent *new* discoveries and is reset each time a *new* neighbor is found. If no new neighbors are discovered within a given period, the unit will timeout and abort the search. The effect of this new timeout mechanism is studied in Part C of the next section.

## V. Performance Evaluation

We test various device discovery methods under different link models based on Bluehoc [2], which is a Bluetooth extension of NS2 developed by IBM. Enhancements are made as follows. First, we endow each node with both master and slave capabilities by unifying their data structure. Devices can thus switch their roles freely during device discovery, i.e., from inquirer to scanner or vice versa. Second, within the allowance of specification, each device carries out device discovery with full capacity prior to the topology formation. Last, we extend the asymmetric link model to symmetric ones (in Section III), and implement three mutual proximity awareness methods (in Section IV). Note that our implementation does not modify specification, e.g., extend the ID format or shorten the backoff timer. So the following simulation results are fully specification compatible.

### A. Experiment setting

Experiments run on three generated *visible graphs* shown in Fig. 4, representing sparse, medium dense and dense scenarios. Each line in the topology denotes a bi-directional link that *could be* established. There are 7, 24, and 63 potential bidirectional links for Topologies 1, 2 and 3 respectively. We consider an asynchronous case in which each device randomly starts to search for other devices in the first two seconds. To remove the statistical error, each point of simulation data is averaged over 50 independent runs. Extensive simulations are done with different variations of the device discovery algorithms. Due to the limited space, only a representative subset of the results is presented below. As device discovery methods are independent of the underneath symmetric link models, we use Model 1 as default in the following

experiments. In Table I, parameters used in our experiment are compared with the default setting in Bluehoc. More details can be found in [10].

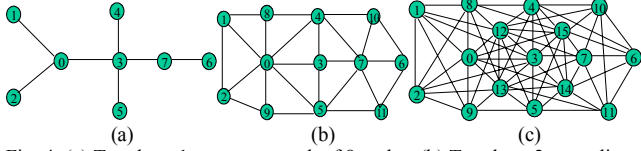


Fig. 4: (a) Topology 1, a sparse graph of 8 nodes. (b) Topology 2, a medium dense graph of 12 nodes. (c) Topology 3, a dense graph of 16 nodes

TABLE I: Parameters Used in Bluehoc and Our Experiments

Parameters	Bluehoc	Experiments	Parameter description
Period_Low	NULL	11.25	Lower bound of the state residence time
Period_High	NULL	Via Sim.	Upper bound of the state residence time
Period_Fix	NULL	Via Sim.	Fixed state residence time in Model 2
State_Prob	NULL	0.5	State transition Prob.
InquiryTO	10.24s	Via Sim.	Device discovery timer
InqRespTO	1s	640ms	Inquiry response timer
Ninq/Npage	256/128	1	Train repetition size
PageTO	2.56s	22.5ms	Page timer
PageScanTO	NULL	22.5ms	Page scan timer
PageRespTO	11.25ms	11.25ms	Page response timer
NewConTO	11.25ms	11.25ms	New connection timer

## B. Determine the upper bound of state residence time

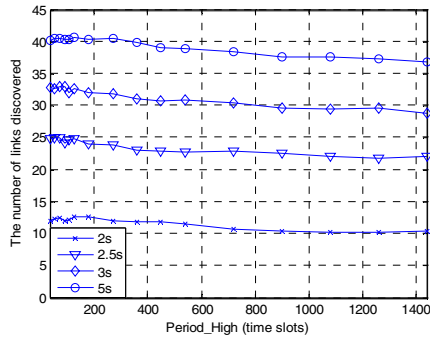


Fig. 5: Method 1, the effect of varying Period\_High in Topology 2

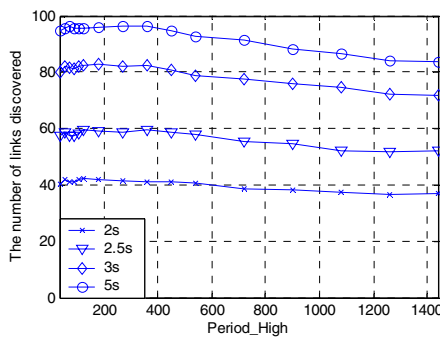


Fig. 6: Method 1, the effect of varying Period\_High in Topology 3

We first examine the parameter setting for point-to-point symmetric link model using Method 1. Since Topology 1 is sparse, the discovery of all the potential links could complete in a short time. We thus focus on Topologies 2 & 3. The lower bound (Period Low) on state residence time is fixed at 11.25ms, or 18 time slots (of slot size 625 $\mu$ s). Figs. 5 & 6 show the effect of varying the state residence time upper bound (Period\_High) on the total number of links discovered at

different time instants (namely, 2s, 2.5s, 3s, and 5s). It can be seen that the performance gradually deteriorates as Period\_High increases. Further results (including those obtained by Methods 2 & 3 [10]) show that near-optimal performance can be achieved around Period\_High=126 time slots. This value is then adopted in the following simulations.

## C. Determine the discovery timeout value

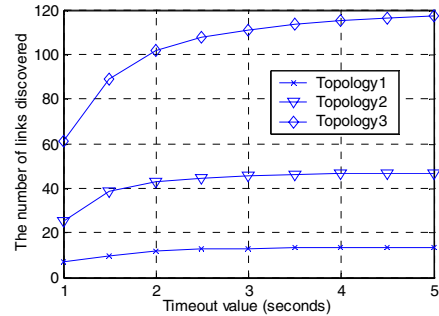


Fig. 7: Method 3, the effect of timeout on the discovered links

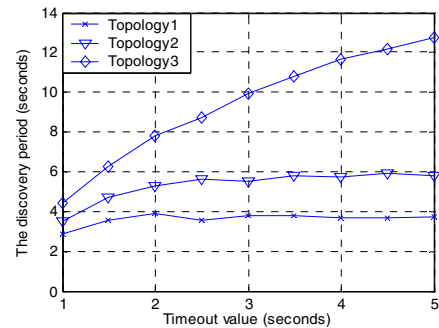


Fig. 8: Method 3, the effect of timeout on the discovery period

Based on our proposed relative timeout mechanism, Figs. 7-8 study the effect of varying timeout values on the performance of device discovery Method 3. (Again, similar conclusions can be drawn for other two methods [10].) Similar trend can be observed in both figures. In Fig. 8, the device discovery period generally increases as the timeout value increases. Notably, the increase for Topology 3 is the highest. This is because in less dense topologies, a small timeout value is enough to find all potential links. We thus set the discovery timeout timer at a value that allows 90% of the links to be discovered even in the densest topology. This corresponds to a relative timer of 4s for Method 1, and 3s for Methods 2 & 3. They are then adopted in the remaining simulations.

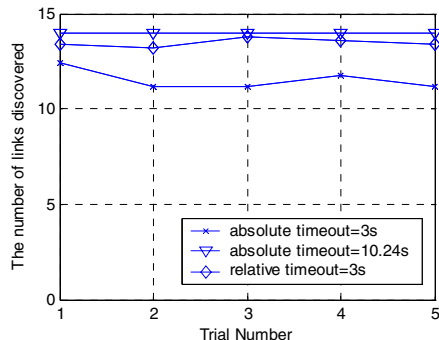


Fig. 9: links discovered with different timeout schemes in Topology 1

Figs. 9-12 compare the mechanism of relative timeout with fixed timeout over sparse and dense topologies. It can be seen that an appropriate value, say 3s, for the relative timer can

accommodate almost all the scenarios. But a fixed timer is only suitable for some specific geographical settings. For example, a fixed timer of 10.24s is suitable for the dense scenario in Topology 3 but its performance deteriorates in Topology 1. Reversely, a fixed timer of 3s has acceptable performance in Topology 1 but poor in Topology 3.

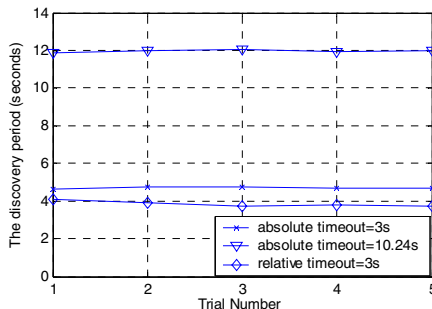


Fig. 10: discovery period with different timeout schemes in Topology 1

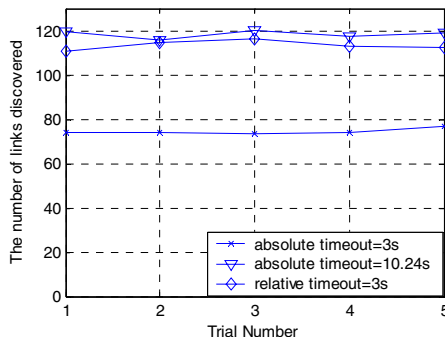


Fig. 11: links discovered with different timeout schemes in Topology 3

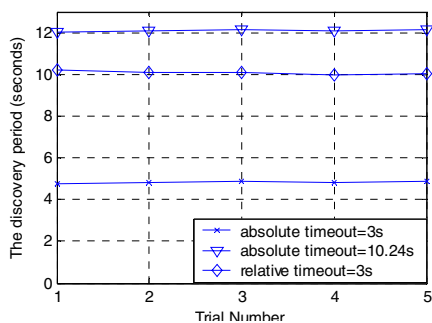


Fig. 12: discovery period with different timeout schemes in Topology 3

#### D. Compare different device discovery methods

The three device discovery methods proposed in Section IV employ different mechanisms to achieve proximity awareness. In Figs. 13-14, their performances, measured by the number of links discovered and the length of the discovery period, are compared using the same three topologies in Fig. 4. We can observe that all three methods give comparable performance when the topology density is sparse (i.e. Topology 1), but their performance gap increases as the topology density increases. Both Methods 2 & 3 excel Method 1 due to the fast operation of page handshake. As expected, Method 3 gives the best performance in the sense that it could find all the links in Topologies 1&2 within 4 and 6 seconds respectively, and find over 90% links in Topology 3 within 10 seconds. This is because in dense networks repetitive handshake occurs frequently, the effort of Method 3 in minimizing page handshake redundancy pays off.

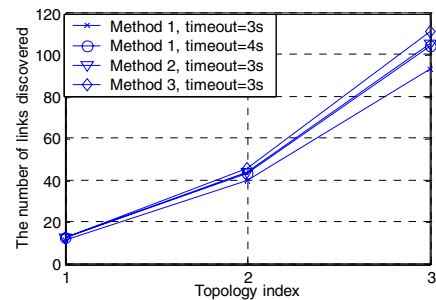


Fig. 13: links discovered using different methods

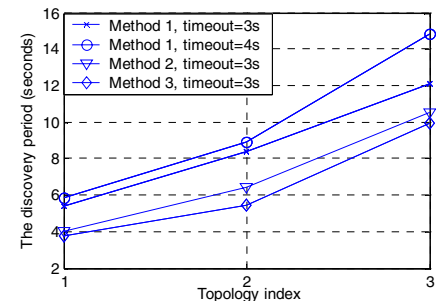


Fig. 14: discovery period using different methods

#### VI Conclusions

In the paper, we studied the issue of device discovery in Bluetooth system. Three point-to-point symmetric link models were compared analytically. To obtain the proximity awareness among a group of devices, three control information exchange methods were also proposed. Combined with a new relative timeout mechanism, substantial performance enhancement was observed. Simulations showed that our proposed solutions can find over 90% links within 10 seconds in a dense topology with average nodal degree of 8. Unlike some existing approaches, it is fully compliant with the current Bluetooth Specification 1.2.

#### References

- [1] Bluetooth Special Interest Group, "Specification of the Bluetooth System 1.2", [www.bluetooth.com](http://www.bluetooth.com)
- [2] Bluetooth Extension for ns (Bluehoc), [oss.software.ibm.com/developerworks/opensource/bluehoc/](http://oss.software.ibm.com/developerworks/opensource/bluehoc/)
- [3] T. Salonidis, P. Bhagwat, and L. Tassiulas, "Proximity awareness and fast connection establishment in Bluetooth," *MobiHOC*, 11 Aug. 2000, pp. 141–142.
- [4] T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire, "Distributed topology construction of Bluetooth personal area networks," *INFOCOM* 2001.
- [5] L. Yong, M.J. Lee, and T.N. Saadawi, "A Bluetooth scatternet-route structure for multihop ad hoc networks," *IEEE Journ. Sel. Areas in Comm.*, Volume: 21 Issue: 2, Feb 2003
- [6] C. Petrioli, S. Basagni, and M. Chlamtac, "Configuring BlueStars: multihop scatternet formation for Bluetooth networks," *IEEE Transactions on Computers*, Volume: 52 Issue: 6, June 2003, pp. 779–790
- [7] S. Basagni, R. Bruno, and C. Petrioli, "A performance comparison of scatternet formation protocols for networks of bluetooth devices," *Pervasive Computing and Comm.*, 2003.
- [8] C. Law and K.Y. Siu, "A new Bluetooth scatternet formation protocol," *Kluwer Journ. of Mobile Net. and App.*, Vol 8, pp. 485-498, Oct. 2003.
- [9] R.W. Woodings, D.D. Joos, T. Clifton, and C.D. Knutson, "Rapid heterogeneous ad hoc connection establishment: accelerating Bluetooth inquiry using IrDA," *WCNC2002*.
- [10] Changlei Liu, "Bluetooth Network Design", MPhil. thesis, The University of Hong Kong, 2003. ([www.eee.hku.hk/~kyeung/clliu\\_thesis.pdf](http://www.eee.hku.hk/~kyeung/clliu_thesis.pdf))