A Simple and Fast Algorithm for Bluetooth Network Formation

Yuh-Jzer Joung[1] and Geng-Dian Hwang[2]

[1] Dept. of Information Management
joung@ntu.edu.tw
[2] Dept. of Electrical Engineering,
National Taiwan University, Taipei, Taiwan
view@iis.sinica.edu.tw

1 Introduction

We present a simple and fast distributed algorithm for Bluetooth network formation. Our algorithm requires no knowledge on the underlying physical topology, and proceeds in only one phase to connect the devices. If the physical topology is weakly connected, then the algorithm guarantees that the resulting network will be connected with high probability. The algorithm is also symmetric in the sense that each device executes exactly the same algorithm, thereby increasing the flexibility of network deployment. In contrast, previous results (e.g., [6, 3, 2, 5,10,1, 8, 9, 4]) on Bluetooth network formation require the underlying physical topology to be a complete graph, nodes to be asymmetric, or proceed in more than one phase to connect disconnect components constructed in the first phase.

2 The BlueTag Algorithm

A distinguishing characteristic of Bluetooth link formation procedure is that two devices must be in a pair of complementary states—*INQUIRY* and *INQUIRY SCAN*. The two devices hop through the same, predetermined, frequency sequence, but the one in INQUIRY state hops at a faster rate than the other so that they have a chance to 'meet' in the same frequency, and therefore to discover each other and exchange the necessary information for synchronization.

Existing algorithms either let Bluetooth devices randomly alternate between the two complementary states, or predetermine devices' states for them to stay throughout the discovery process. Our approach is to let each node alternate between INQUIRY and INQUIRY_SCAN in some pattern determined uniquely by its *root* value, initialized to its device ID *BD_ADDR*. We refer to these patterns as *state alternation sequences*. Nodes with the same *root* value have exactly the same, and *synchronized*, sequence; while nodes with different *root* values have different sequences. This design allows two nodes with different *root* values to be in complementary states, and therefore have a chance to discover each other

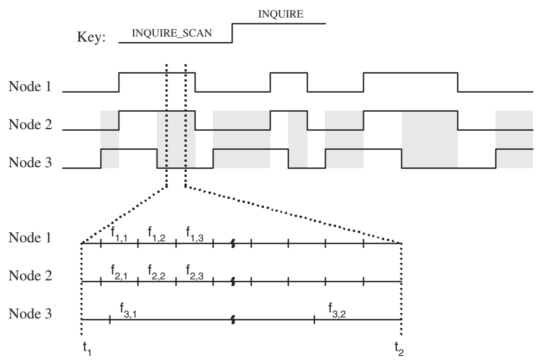

Fig. 1. State alternation sequences. The expanded interval shows the hopping frequencies

via the baseband. On the other hand, nodes with the same *root* value are always either all in state INQUIRY, or all in INQUIRY_SCAN, and so have no chance to discover one another. For example, in Figure 1, nodes 2 and 3 are in different states in the gray intervals, while nodes 1 and 2 are always in the same state because they have the same synchronized sequences.

The algorithm is outlined in Figure 2. We refer to the algorithm as BlueTag. Lines 3-7 initialize variables. In line 6, the initial state is INQUIRY; but this can be set arbitrarily. The timer *ALT_TIMER* is used to determine when to alternate state (INQUIRY or INQUIRY SCAN). After initialization, every node alternates between INQUIRY and INQUIRY SCAN according to its state alternation sequence determined by its current *root* value. When *ALT_TIMER* is timed out (lines 9-12), u computes the time to stay in the new state and then alternates to the state. State alternation sequences can be implemented by some pseudorandom number generators using a device's *BD_ADDR* as seed.

Meanwhile, if some neighbor v is discovered by u (line 13), they establish a communication link and exchange information for adjusting their state alternation sequences (line 15). Then, the one with smaller *root* value adjusts its *ALT_TIMER* and *state* to synchronize with the other. Both nodes reset their *DC_TIMER* to discover other nodes with different *root* values. We can say that a temporary piconet is formed by u and v to exchange information for synchronization (other than to know the existence of each other). However, the piconet is broken after the procedure. The two nodes then continue on their own to discover other nodes.

DC_TIMEOUT can be determined by analyzing the connection establishment time T_c between two Bluetooth devices with different state alternation sequences, and then use the following empirical formula [7]:

$$DC\_TIMEOUT = E[T_c] + \sqrt{Var[T_c]} + r_{\max} \tag{1}$$

A node's *DC_TIMER* will time out when it cannot find any neighboring node with a different *root* value. Then it terminates its action in the network formation

```
Main(Bluetooth node $u$)
{
    $root \leftarrow u.BD\_ADDR$
    reset $DC\_TIMER$
    $neighbors \leftarrow \emptyset$
    $state \leftarrow$ INQUIRY
    $ALT\_TIMER \leftarrow -1$
    /* beginning of neighbor discovery process */
    while $DC\_TIMER$ not timed out {
        if $ALT\_TIMER$ timed out {
            compute new $ALT\_TIMER$
            $state \leftarrow \overline{state}$ /* alternate state */
        }
        if a Bluetooth node $v$ is discovered {
            $neighbors \leftarrow neighbors \bigcup \{v\}$
            $EXCHANGE\_INFO(v)$
            if $v.root > u.root$ {
                $u.root \leftarrow v.root$
                /* synchronize with $v$ */
                compute new $ALT\_TIMER$
                $state \leftarrow \overline{state}$ /* change to $v$'s state */
                }
            reset $DC\_TIMER$
        }
    }
    $ROLE\_ASSIGNMENT\ (u, neighbors)$
}
```

Fig. 2. Outline of the algorithm executed by a node u

process. Note that at this point although the overall network construction may not yet complete, the terminating node has full knowledge on its neighboring nodes in the network to be constructed. So the node may now start its upper-level application. If in the application the node needs to communicate to its neighbors and they have not yet started the application, the communicating messages can be queued until they are ready. So a node will not necessarily be blocked from starting the application even if other nodes may still be in the network formation process.

To illustrate the algorithm, consider Figure 3. We label each node with $i[j]$, where i is the node's *BD_ADDR*, and j is its *root* value. Part (a) shows the initial physical topology, where dashed lines represent physical links. In (b), node 3 and 1 discover each other and establish a link (solid line). Since node 3 has larger *root* value, it beats node 1, and makes node 1 to carry its *root* value, so as to synchronize with its state alternation sequence. In (c), 4 and 5 discover each other and establish a link. At the same time, 2 and 3 discover each other and establish a link. Moreover, node 1 terminates the algorithm (represented in gray color) because it cannot discover any node. In (d), 5 and 3 discover each

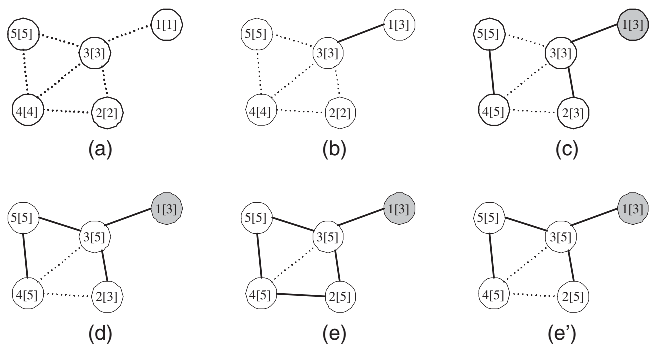

Fig. 3. A scenario of BlueTag

other and establish a link. Node 3 (with new *root* 5) now synchronizes with node 5, and so its synchronization with node 2 (still with *root* 3) is broken. Note that the previously established link between 2 and 3 (stored in their variables *neighbors*) is not affected even though each node re-synchronizes with another. In (e), 4 and 2 discover each other and establish a link. Node 2 now carries *root* value 5. So nodes 2 and 3 are now synchronized again (by *root* value 5 this time). A variation of scenario (e) (shown in (e′)) is that, instead of 4 and 2 discovering each other, 3 and 2 re-discover each other as they carry different *root* values after (d). If this is the case, then no new link will be established, as 3 and 2 already knew each other from scenario (c). However, node 2 will adjust its *root* to 5, avoiding it from re-discovering 3, or discovering 4. In either case, nodes $2, 3, 4$ and 5 will all terminate because they cannot discover any more node. From the example we can also see that the nodes may not terminate with the same *root* value (but the connected property of the final network is guaranteed).

To complete the network formation, each node needs to know which piconets it is involved in and its roles in them. This can be done by either *root-based*—x is the master of y if x has larger root value than y has when they last discover each other; *ID-based*—x is the master of y if x has larger *BD_ADDR* than y has; or *Random*—x is the master of y if x is in INQUIRY state when they last discover each other.

References

1. S. Basagni and C. Petrioli. Multihop scatternet formation for bluetooth networks. VTC Spring 2002.
2. C. Law, A. K. Mehta, and K.-Y. Siu. Performance of a new bluetooth scatternet formation protocol. ACM MobiHoc 2001.
3. C. Law and K.-Y. Siu. A bluetooth scatternet formation algorithm. IEEE Symposium on Ad Hoc Wireless Networks.
4. C. Petrioli and S. Basagni. Degree-constrained multihop scatternet formation for bluetooth networks. IEEE Globecom 2002.

5. L. Ramachandran, M. Kapoor, A. Sarkar, and A. Aggarwal. Clustering algorithms for wireless ad hoc networks. 4th international workshop on Discrete algorithms and methods for mobile computing and communications, 2000.
6. T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire. Distributed topology construction of bluetooth personal area networks. INFOCOM 2001.
7. T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire. Proximity awareness and ad hoc network establishment in bluetooth. Technical Research Report CSHCN 2001-1 (ISR TR 2001-10), University of Maryland, 2001.
8. I. Stojmenović. Dominating set based bluetooth scatternet formation with localized maintenance. IPTPS 2002.
9. Z. Wang, R. J. Thomas, and Z. Haas. Bluenet – a new scatternet formation scheme. HICSS-35, 2002.
10. G. V. Záruba, S. Basagni, and I. Chlamtac. Bluetrees-scatternet formation to enable bluetooth-based ad hoc networks. ICC 2001.