

Blueline: A Distributed Bluetooth Scatternet Formation and Routing Algorithm

RUAY-SHIUNG CHANG AND MING-TE CHOU

Department of Computer Science and Information Engineering

National Dong Hwa University

Hualien, 974 Taiwan

E-mail: rschang@mail.ndhu.edu.tw

The emerging Bluetooth technology is a best-known PAN (Personal Area Networks) technology. It still has some issues left open in the current specification. Among them, the scatternet formation and routing are two major issues. In this paper, we proposed a new Bluetooth scatternet formation algorithm and its routing algorithm. Our method constructs and maintains a scatternet in a distributed way and does not need all nodes to be in the transmission range of each other. We use the PARK mode to let Bluetooth devices have more chances to link each other in order to build a better-connected scatternet. We also propose an on demand routing algorithm for the scatternet constructed. Experimental results show that the proposed algorithms are quite efficient and effective.

Keywords: bluetooth, routing algorithm, personal area networks, scatternet, piconet

1. INTRODUCTION

In our daily lives, handheld devices are becoming more important. Most people have already carried cell phones, digital cameras, audio and video players, laptop computers or health monitoring devices with them wherever they go. They need to connect these handheld devices with desktops to synchronize data or create backups. In most cases, these handheld devices do not have compatible data communication interfaces. Even if they do, they must be connected with cables and be configured correctly. This is not very convenient. One simple solution for this mess is doing away with all cables and instead uses the short-range wireless technology for connectivity. Short-range wireless connections have a set of competing technologies. They are usually for indoor use and with very short distance. Short-range wireless technologies provide a cableless connection between portable devices. They may also provide Internet connection services for users by accessing a wireless node that has wired Internet connection.

Due to its intended pervasive use, short-range wireless communication technology usually focuses on low cost and low power usage. One of the best-known such technologies is Bluetooth [6, 15]. The basic communication structure in a Bluetooth network is a piconet. It has a star-shaped topology. Bluetooth specification defines a multihop ad hoc network structure, called a scatternet, to interconnect overlaying piconets. Fig. 1 is an

Received January 15, 2004; revised April 20, 2004; accepted June 14, 2004.

Communicated by Yu-Chee Tseng.

* This research was supported in part by R.O.C. NSC under contract number 92-2219-E-259-001 and 92-2213-E-259-020.

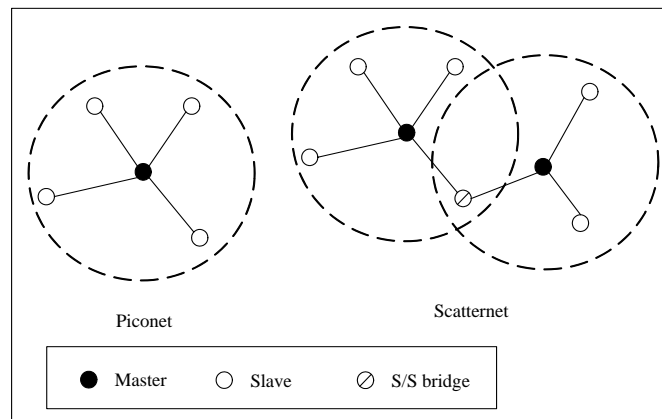


Fig. 1. A piconet (left) and a scatternet (right).

example of piconet and scatternet. However, the scatternet formation problem is left open in the current Bluetooth specification.

The connection status of a Bluetooth node has three states: Standby state, Intermediate State, and Connection state. Depending on the connection state and the need to transit between states, a node can stay at one of the following seven sub-states: Inquiry, Inquiry Scan, Inquiry Response, Page, Page Scan, Page Response, and Master Response. After joining a piconet, a slave is at one of the four modes: Active, Sniff, Hold, and Park [15]. Bluetooth nodes move in and out of these states, sub-states, and modes through commands from the Bluetooth link manager or from internal signals in the link controller.

In any Bluetooth network, at the beginning a Bluetooth node is not connected to any other node. They will be in the standby state. In this state, they do not transmit any data, but only listen to the hop channels with a very low duty cycle. Bluetooth uses a procedure known as inquiry for discovering other nodes, and uses another procedure known as page to establish connections. Both are asymmetric procedures. It means that the inquirer and the inquired nodes perform different actions (inquiry and inquiry scan, page and page scan). In other words, two nodes that want to connect with each other need to start from a different initial state. After two nodes are connected, one will act as the master and the other will be a slave. The master will control the information transfer between itself and all slaves. A Bluetooth slave node will be in one of the four modes mentioned above at one time. Besides Active mode, the other three modes are power-saving modes. When a piconet is operational, a slave keeps the power on to communicate with the master node. In order to increase the battery life, it is possible to switch a slave into a power-saving mode. A slave in the power-saving mode sleeps most of the time and wakes up only occasionally or periodically.

When in the active mode, a slave obtains a 3-bit active member address (AM_ADDR) from the master. It represents the shorter identity of an active slave in the piconet compared to the longer Bluetooth identity (BD_ADDR). When the master sends a packet out, it will use this AM_ADDR to indicate the destination. If the AM_ADDR is 0, it means a broadcast message in the piconet. Since AM_ADDR has 3 bits, it implies that the maximum number of active slaves in a piconet will be 7.

Park mode is a power-saving mode with very little slave activity. In this mode, a slave will give up the AM_ADDR. Two new addresses, which are received from the master, will be used in the park mode. One of the new addresses is an 8-bit parked member address (PM_ADDR). The other is an 8-bit access request address (AR_ADDR). The PM_ADDR is used in the master-initiated unparking procedure. The master can wake up a parked slave to activation by sending the unpark message with PM_ADDR of the selected slave. The AR_ADDR is used by the slave-initiated unpark procedure. If a parked slave wants to return to active mode, it sends the AR_ADDR to the master in the beacon channel to request a slave-to-master time slot. The master will return a control message to unpark it. In addition, the master can also wake up a parked node with the 48-bit BD_ADDR by setting all bits of PM_ADDR to zero. The all-zero PM_ADDR address has a special usage. If a parked node has an all-zero PM_ADDR, it can only be unparked by using the BD_ADDR. It implies that a piconet can have as many number of park nodes as the BD-ADDR allows. This parking/unparking mechanism allows the master to manage more than seven slaves in its piconet.

In this paper, we devise a distributed Bluetooth scatternet formation algorithm using this parking property. When a master slave pair is formed, the slave is immediately parked such that the master will not be limited by already having seven active slaves. This method is simple and effective and is compatible with current Bluetooth specification. Furthermore, to reduce the time and path length in routing, we want two Bluetooth nodes to communicate directly if they are within each other's transmission range. The use of park mode allows us to do this without the limitations of the number of active slaves that a master can have. Since a straight line is the shortest way to connect to points in space, we will call our algorithm **Blueline** to indicate that the communicating path between two Bluetooth nodes is shorter compared to other scatternets.

The rest of this paper is organized as follows. Section 2 gives an overview of the related work for Bluetooth scatternet formation in the literature. Section 3 describes the proposed scatternet formation algorithm. Performance evaluation results are presented and analyzed in section 4. Section 5 concludes this paper.

2. RELATED WORK

Because the detailed scatternet formation algorithm was left open by the Bluetooth specification, it is an interesting research topic. We briefly review some of the algorithms proposed in the past.

Lin *et al.* [7] propose an algorithm to form a scatternet called "BlueRing". A BlueRing is a ring of nodes where each node is itself a piconet. A ring transmission direction is imposed. In each node (piconet), there are two bridges connecting upstream and downstream piconets in the ring. Assuming each node is within the radio coverage of one another, a centralized scatternet formation algorithm is used. The scatternet construction consists of two stages. In the first stage, a leader is selected by choosing the node that has gathered the largest information base from other nodes. In the second stage, the leader designates nodes to become masters according to the desired ring topology. The scatternet formed is more scaleable and fault tolerant.

Zaruba *et al.* [14] introduce a scatternet formation algorithm called "Bluetree". This

algorithm starts formatting scatternet from a specific node called “Blueroot”. The Blueroot plays a master role and connects to its neighbor nodes that have not been connected by other nodes. These nodes will be the Blueroot’s slaves. When these nodes become the slaves of Blueroot, they will change their roles to master and try to connect to other neighbor nodes that have not joined any piconet. This procedure is recursively repeated till all nodes are connected. Bluetree can also speed up the scatternet formation process by selecting more than one root and then merging the trees generated by each root.

Sun *et al.* [12] also introduce a scatternet formation algorithm. Their concept comes from binary search tree. In their algorithm, a Bluetooth node has five modes. They are inquiry mode, inquiry scan mode, page mode, page scan mode and connection mode. Their formation steps are like B-tree. Each node has a range value. This range value indicates the range of BD_ADDR to which Bluetooth nodes can connect. When a new node wants to join a piconet, the root node will search the range values of children. The smallest numbered node whose value is larger than the new node will be the parent of this new node. Otherwise, the root node will be the parent of this new node.

Tan *et al.* [13] introduce a scatternet formation algorithm called TSF (Tree Scatternet Formation). It is also a tree-based scatternet formation algorithm. Each node is free initially. It has three rules to form a scatternet. They are:

1. Free nodes may only connect to other free nodes, or to non-root nodes.
2. Root nodes may only connect to other root nodes.
3. Non-root nodes do not attempt to form larger scatternet with nodes that are not free nodes.

Sato *et al.* [11] introduce a scatternet formation algorithm with the following two conditions.

1. A node can belong to up to two piconets.
2. Two piconets share one node at most.

By these two conditions, their algorithm will form a tree-like scatternet. At first, each node finds out the neighbor nodes by neighbor discovery protocol. When the neighbor discovery is finished, it will enter the piconet formation phase to form piconets. The last phase is disconnection detection phase. This phase will check its connection information. If there is a neighbor not in the connect list, it will ask a slave node to reenter the piconet formation phase.

Law *et al.* [5] partition Bluetooth devices into components. A component is a set of interconnected devices. It might be a node, a piconet or a scatternet. There is one leader in each component. Only leader node can merge other components. At first, all nodes are leader nodes. They try to connect to another node. When a link is established, the master node will be the leader of this component. It will try to connect and merge other components. Finally, all components will be connected to a big scatternet.

Salonidis *et al.* [10] introduce a scatternet formation algorithm called BTCP (Bluetooth Topology Construction Protocol). It is a leader-election type scatternet formation algorithm. It contains two phases. In phase one, each node tries to connect to other nodes.

When a link is established, the one with smaller BD_ADDR wins and collects the address and clock information from the loser. It then tears down the links and tries to connect with another. The loser stays in page scan mode for paging. Eventually, the node with the smallest address becomes the leader and collects the address and clock information of all nodes. In the second phase, the leader uses this information and a centralized algorithm to form a scatternet.

Baatz *et al.* [1] devise a scatternet formation algorithm based on BTCP. In BTCP, the leader collects the address and clock information of all nodes. It implies that the leader can form any topology of scatternet as it wants. This algorithm uses the factor graph to form its topology. It chooses several 1-factor graphs, which make all nodes getting connected.

Petrioli *et al.* [9] introduce a scatternet formation algorithm called BlueStar. Their algorithm proceeds in three phases. At the first phase, nodes discover neighbor nodes by their discovery protocol. The next phase takes care of BlueStar (piconet) formation. The final phase concerns the selection of gateway devices to connect multiple BlueStars.

Liu *et al.* [8] build a scatternet on the fly with the routing in minds. They build scatternet only along the multihop routes with traffic demands. All nodes are not in connection mode when they do not have traffic. When they want to transmit to others, they will build a temporary routing path.

The common parts of most previous researches assume that nodes do not move or are within the communication range of each other to form fully connected scatternet. Besides this, the resulting scatternet topology is mostly tree-like. The tree-based scatternet topologies have a minimum number of piconets. However, they have longer routing paths with limited robustness and efficiency. Furthermore, since there is only one path between any two nodes in a tree, the routing between two Bluetooth nodes is relatively simple in tree-like scatternets.

3. THE BLUELINE ALGORITHM

3.1 The Scatternet Formation Algorithm

Unlike most other wireless technologies with connectionless data link layer, Bluetooth requires explicit link formation and channel maintenance between communication devices. In order to conserve power, the physical channel and data link can only be created and supported when Bluetooth node really wants to transmit. If there is no link established, Bluetooth device will go to a low power "Standby" state. Otherwise, it will go to the "Connection" state. The "Connection" state has four submodes. They are "Active", "Sniff", "Hold" and "Park". Among these four modes, the "Sniff", "Hold" and "Park" are power saving mode. The low power consumption is an important feature of Bluetooth. However, most scatternet formation algorithms ignored this power saving principle. They keep Bluetooth devices in "Active" mode that has the worst power consumption.

Most of current Bluetooth scatternet formation algorithms result in some specified topologies. The data routing path will depend on the topologies. For two neighboring nodes, their data routing path might be through several hops even though the shortest path between them is a direct connection. Our proposed scatternet formation algorithm

will allow two Bluetooth nodes to form a connection and communicate directly if they are within each other's transmission range. The purpose is to form a topology with the minimum number of hops for routes.

The "Park" mode in Bluetooth can support a large number of Bluetooth nodes in a piconet. A Bluetooth device in "Park" mode also has more free time slots for inquiry or inquiry scan than other devices. If a Bluetooth device is successful in inquiry to another, it will play the master role of this newly formed piconet. The other will be a slave. If the master node parks all its slaves when they finished the action of building connections, all nodes of this piconet will have free time slots to join another piconet or search new neighbor nodes and invite them to join. This is the idea behind our proposed scatternet formation algorithm – Blueline.

Speaking clearly, Bluetooth nodes randomly choose inquiry or inquiry scan for their initial states. Each node is allowed to alternate for a predefined device discovery duration between inquiry mode and inquiry scan mode. It remains in each mode for a time selected randomly and uniformly in a given time range. It is called a round-time. If a Bluetooth node's initial state is inquiry, it searches for the other nodes that is in its communication range and invites them to join its piconet. Otherwise, it waits for its neighbors' invitation to join their piconets. If the round-time expires, a node will randomly select its next inquiry mode. When there are two nodes in opposite inquiry modes and they make a handshake, a master-slave connection and hence a piconet will be formed. After building this piconet, the master node will have the slave parked. Both of them will keep their previous state (inquiry and inquiry scan) and continue to discovery neighbors until this round is ended. They alternate between the opposite inquiry modes until the discovery duration time is over.

The pseudo-code for Blueline running at each node is shown below. CLKN is the native counting clock of Bluetooth node. It increases constantly with time. DISCOVER_TIME is the total time running Blueline algorithm. SWITCH_BASE and SWITCH_INCR are the basic running time and the maximum random increment time for each round. α is the probability of selecting inquiry scan mode in each round.

```

Procedure Blueline() {
  discover_timeout_timer = CLKN + DISCOVER_TIME /* Define the total running time. */
  while discover_timeout_time > CLKN /* If the interval clock is still less than the */
  { x = a random number in [0, 1) /* preset timer, enter another round. */
    round_interval = SWITCH_BASE + x * SWITCH_INCR /* Define the time interval for */
    round_timeout_timer = CLKN + round_interval /* this round. */
    x = a random number in [0, 1]
    if x <  $\alpha$  then /* Determine to act as a possible master or slave. */
      repeat
        Inquiry Scan mode /* slave state */
        if find a new neighbor /* Detect a master. */
          be paged and building a link
          be parked
      until
        CLKN > round_timeout_timer /* This round is ended. */
    else

```

```

repeat
  Inquiry mode          /* master state */
  if find a new neighbor /* Find a slave. */
    page the new neighbor and build a link
    park this new neighbor
until
  CLKN > round_timeout_timer /* This round is ended. */
} /* end of while */
} /* end of Blueline */

```

In the above algorithm, an interesting problem is to determine a suitable DISCOVER_TIME and SWITCH_BASE. In [10], the Bluetooth link formation delay R can be approximated by $2 * F + B$, where F is the frequency synchronization delay and B is the random backoff delay. F is at most 20 ms for a 32-hop system and B is at most 639.375 ms. If there are n Bluetooth nodes and assume all nodes are within one another's transmission range, there are $n(n - 1)/2$ possible links. In each round, at most $n/2$ links can be established. Therefore, $(n - 1)$ rounds are needed. As a result, a rule of thumb is to set SWITCH_BASE to be R and DISCOVER_TIME to be $k * (n - 1) * R$, where k is a positive integer.

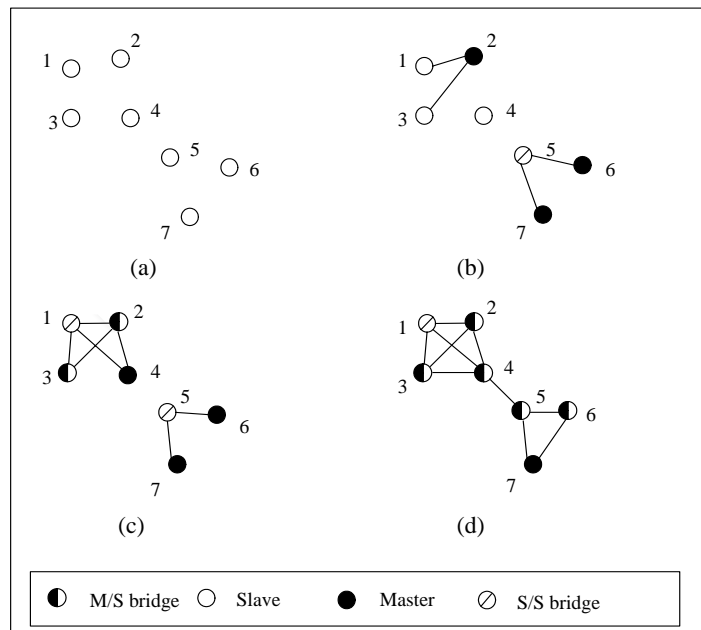


Fig. 2. Blueline scatternet formation.

Fig. 2 is an example of Blueline scatternet formation. Fig. 2 (a) is the initial state. All nodes are in standby mode and start neighbor discovery. Assume that nodes 2, 6 and 7 are in inquiry mode and the others are in inquiry scan mode initially. After a few time

slots gone by, they form three piconets. It is shown in Fig. 2 (b). Nodes 1, 2 and 3 form a piconet. In this piconet, node 2 finds nodes 1 and 3 and invites them to its piconet. Nodes 5, 6 and 5, 7 form their own piconets similarly. In these two piconets, node 5 participates in two piconets simultaneously. Fig. 2 (c) shows nodes 2, 3 and 4 complete their first round and change their states. Node 2 is in inquiry scan and nodes 3 and 4 are in inquiry state. Node 4 finds nodes 1 and 2 and invites them to its piconet. Node 3 finds node 1 and node 1 joins the piconet of node 3. Finally, one of the possible Blueline scatternets is shown in Fig. 2 (d).

The final topology of Blueline is a visibility graph in ideal case. The visibility graph is the network topology where there is a link between any two nodes whose Euclidean distance is not larger than the transmission radius. This has the advantage of reducing the routing path. In the extreme case when all nodes are in the transmission range of each other, their visibility graph is a complete graph. When they want to transmit, they can communicate to each other directly. However, this may lead to more links and more piconets than the tree-like scatternet. If desired, we can reduce the links and piconets in the scatternet by limiting the number of piconets a node can join.

3.2 Finding a Route in Blueline

In the scatternet constructed by Blueline, each node has all of its neighbors' addresses. If it wants to send data to someone in the same piconet, it will be very easy. It just unparks the neighbor or itself if necessary. Then they can begin talking. If the destination is not its neighbor, it will send a route query packet to its neighbors to find the route. This packet can help the sender to find out the shortest path to the destination node. This packet's format is shown in Fig. 3. The packet type indicates the type of this packet. If packet type is 1, then it implies this packet is a route query packet. BD_ADDR of Destination node stores the unique Bluetooth identity of destination node. Hop to live is the number of hops this packet can pass.

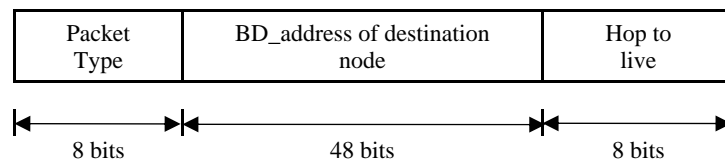


Fig. 3. Route query packet format.

When a node receives a route query packet, it will search its neighbor list for the destination address. If it is not found, hop to live will be decreased by 1. If it is still larger than zero, the node will forward this route query packet to its neighbors and record this query in its query buffer. The query buffer stores the address of this packet sender and the destination address in this packet. If a node receives a route query packet and finds out the destination address in its neighbor list, it will respond a route response packet to the sender. This packet's format is shown in Fig. 4. The packet type field will be 2. The hop number stores the number of hops to the destination node. The first node to return

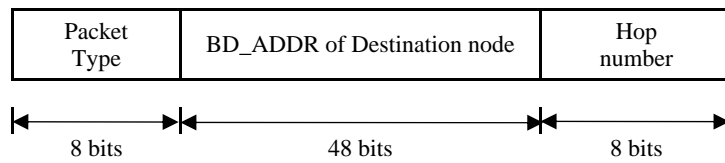


Fig. 4. Route response packet format.

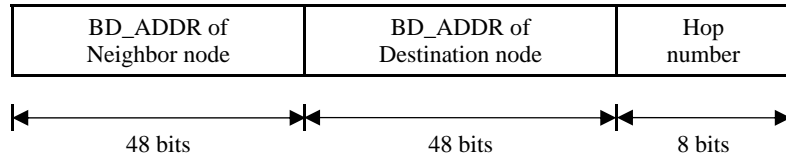


Fig. 5. Routing table cache.

route response packet will have a hop number of 1. If a node receives a route response packet, it will match it to data in the query buffer. If there is a match, hop number will increase by 1. If the sender receives the route response packet, it removes this query from its query buffer and stores this query response information in its cache routing table. The format of cache routing table is shown in Fig. 5. BD_ADDR of Neighbor node is the unique Bluetooth identity of next node for routing. BD_ADDR of Destination node stores the unique Bluetooth identity of destination node. The hop number stores the number of hops to the destination node. If an intermediate node receives the route response packet, it forwards this packet toward the sender and stores this query response information in its cache routing table after removing the query from its query buffer. A node may receive more than one route response packet for the same destination node. It will keep three of them that have the first, the second, and the third smallest number of hops. The pseudo-code for Blueline route discovery protocol is shown below.

```

Procedure send_data() {
  if destination is in neighbor list or route cache
    unpark;
    send data;
  else
    unpark;
    send route request packet to neighbors;
    store the query in query buffer;
    process_receive_packet();
    send_data;
}
Procedure process_receive_packet() {
  if packet_type == 1 /* route query packet */
    if destination is in neighbor list or route cache
      {setup a route response packet with hop number = 1;

```

```

    return route response packet to the sender;}
else
    hop_to_live = hop_to_live - 1;
    if hop_to_live > 0
        forward route request packet to other neighbors;
if packet_type == 2 /* route response packet */
    hop_number = hop_number + 1;
    store route information in route cache;
    find the original request sender in query buffer;
    if original sender is not myself
        forward route response packet to the original sender;
}

```

One thing not described in the above algorithm is the switching policy of a bridge in the scatternet. Basically, we use the first-come-first-serve strategy with time-out mechanism. Once a bridge is sending data in a piconet, it will send out all the data before switching to next requesting piconets. If the current piconet is busy such that the data sending is blocked for a certain time limit, the bridge can then switch to the next piconet.

Another thing to note is the design of the query buffer. For example, in Fig. 2 (d), if both nodes 2 and 3 want to send to node 5, then node 4 should be aware of this situation that only a single route query is needed. This can be handled by the data structure shown in Fig. 6. For each destination, there is only a record. Each new request for the same destination is link-listed. When a query reply comes back, each node in the linked list should be notified of this information.

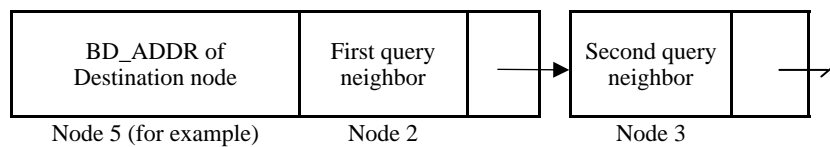


Fig. 6. Query buffer for node 4 when nodes 2 and 3 query node 5 in Fig. 2 (d).

4. SIMULATION

To evaluate the performance of Blueline, we have developed a Bluetooth extension to the VINT project network simulator (“NS2”) [16]. We based our extension on BlueHoc_ex [2, 3], the ns2-based simulator extended by BlueHoc [4]. BlueHoc is an ns2-based simulator released by IBM.

In our simulation scenarios, n Power Class 3 Bluetooth nodes are randomly and uniformly scattered on a geographic area, which is a square of L meters. (The Power Class 3 Bluetooth nodes have the maximum transmission radius of 10 meters.)

Assume one tick is $312.5 \mu s$ and let α be the probability of being in the inquiry scan mode. Fig. 7 shows how the ratio of discovered nodes changes with respect to α . In this

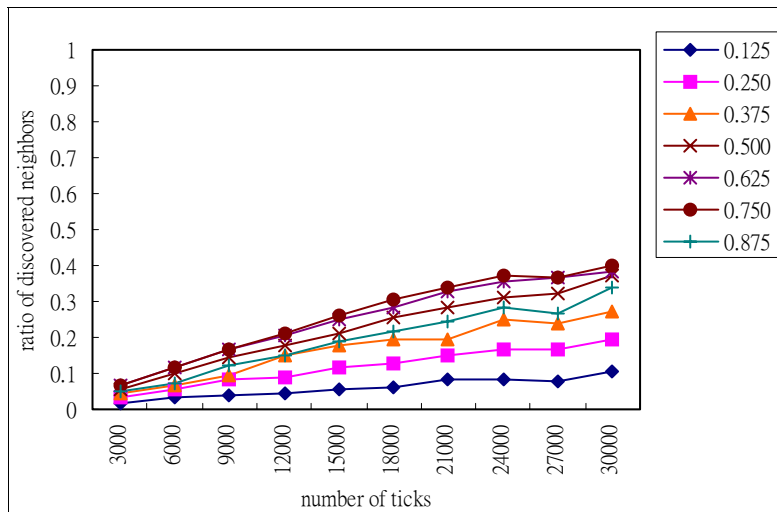


Fig. 7. Effect of different inquiry probability in each round.

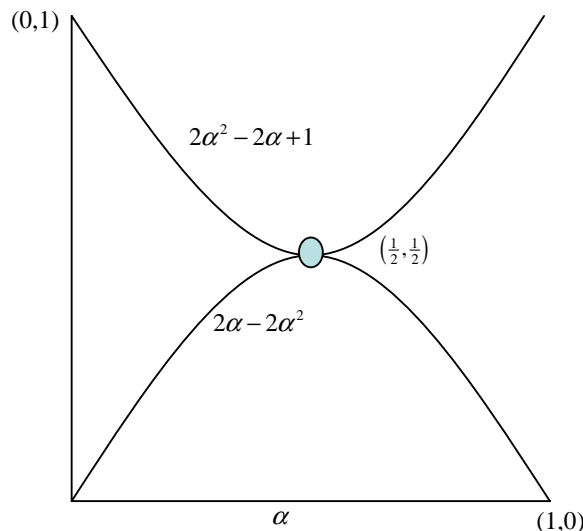


Fig. 8. Matching and unmatched probability curve.

simulation, there are 50 nodes and $L = 20$ meters. x -axis represents the DISCOVER_TIME in the BlueLine algorithm. For two nodes n_1 and n_2 , there are four states for their scanning modes, shown in Table 1. When n_1 and n_2 are in opposite state, there will be a successful master-slave matching with probability $2\alpha(1 - \alpha)$. When n_1 and n_2 are in the same state, the matching will be unsuccessful with probability $\alpha^2 + (1 - \alpha)^2 = 2\alpha^2 - 2\alpha + 1$. The probability curves are shown in Fig. 8. It is obvious that when $\alpha = 1/2$ we have maximum master-slave matching probability. However, from this simulation results, we find that it has better chances of finding neighbors when α is around 0.75. This is

Table 1. Probabilities of different states for two nodes n_1 and n_2 .

n_1 state \ n_2 state	Inquiry mode	Inquiry scan mode
Inquiry mode	$(1 - \alpha)^2$	$(1 - \alpha)\alpha$
Inquiry scan mode	$\alpha(1 - \alpha)$	α^2

because the analysis of $\alpha = 1/2$ only applies to the 1-to-1 case. When there are many Bluetooth nodes, more nodes in inquiry mode (when α approaches 0.75) will let nodes in inquiry scan mode have more chances (since there are more suitors) to be invited to join a piconet. If α increases beyond 0.75, then there will be too few slaves for too many masters.

Fig. 9 shows the effect of different time interval (round_interval in the BlueLine algorithm) for inquiry (inquiry scan) phase. In this figure, intervals of 2000, 4000, and 6000 ticks are tested. Intuitively a larger interval will increase the ratio of discovered neighbors. The simulation conforms to the intuition.

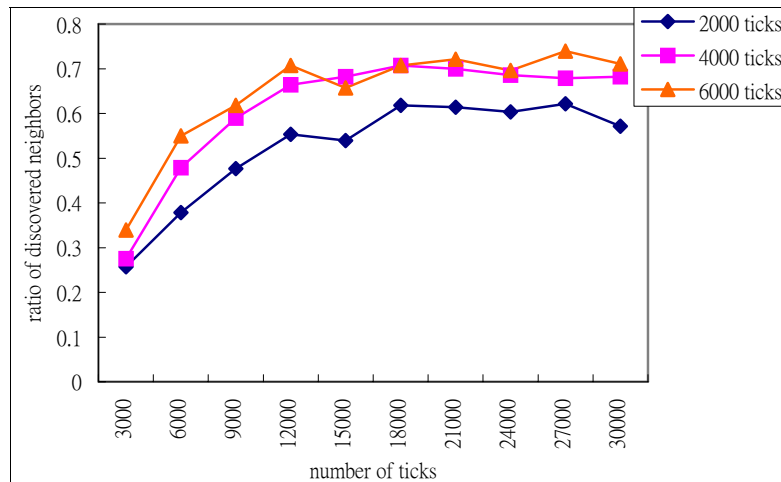


Fig. 9. Effect of different length of round_interval.

Fig. 10 shows the average number of slaves per piconet. From the figure, the number of slaves in a piconet is not much greater than 7. Although we allow each piconet to have as many slaves (in parked mode) as possible, simulation results show that each piconet will not be that crowded. Therefore, the overhead of waking up slave nodes from park mode will not degrade the performance very much.

Fig. 11 shows the average number of piconets in the scatternet. The number of piconets approaches the number of nodes. It means most of the nodes are master/slave bridges. This can save data transmission time compared to slave/slave bridges since a slave/slave bridge needs at least two pollings from two masters for data to pass through it.

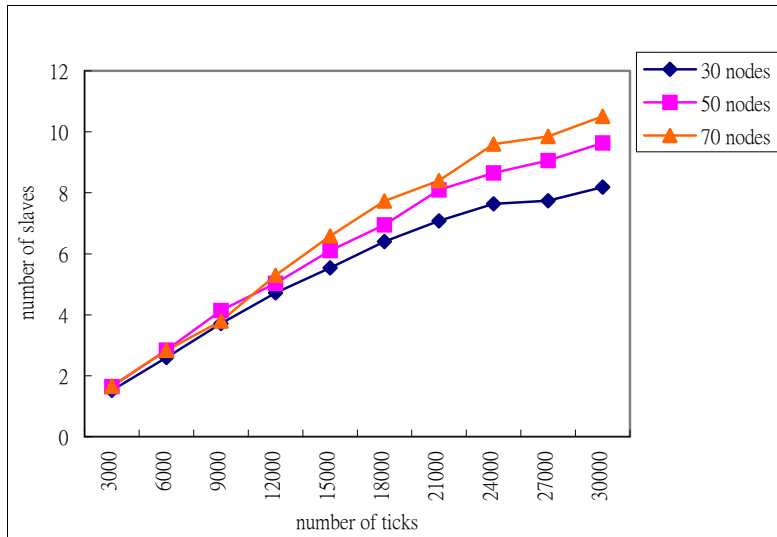


Fig. 10. Number of slaves per piconet.

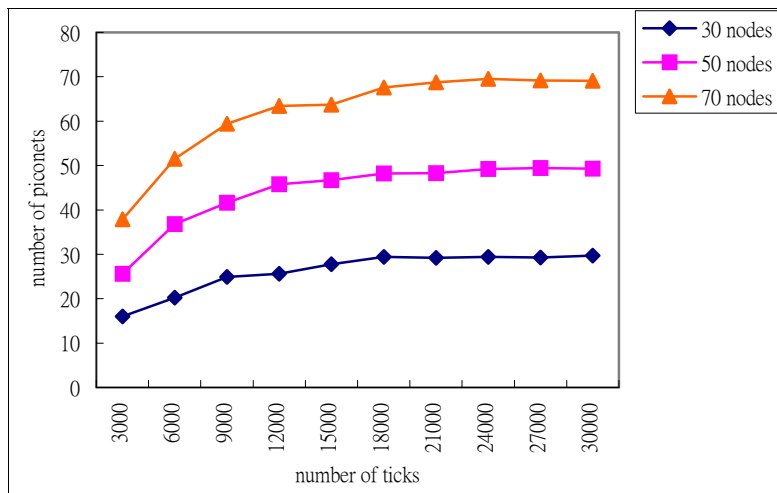


Fig. 11. Number of piconets.

Fig. 12 shows the average number of roles that a Bluetooth device plays in the scatternet. A Bluetooth device in average participates in two piconets in the simulation. Therefore, the overhead from role switching or piconet switching will be limited.

Since BlueLine algorithm emphasizes its richness of connectivities by allowing nodes within communication range to form a piconet using parked slaves if necessary, the length of its routing path would be short. To demonstrate this property, we simulate and compare the average routing paths hops of two other algorithms (Bluestar [9] and Bluetree [14]) with that of BlueLine. The nodes are distributed in an area of 20 * 20 square meters. As can be seen from Fig. 13, BlueLine indeed performs better in reducing routing hops.

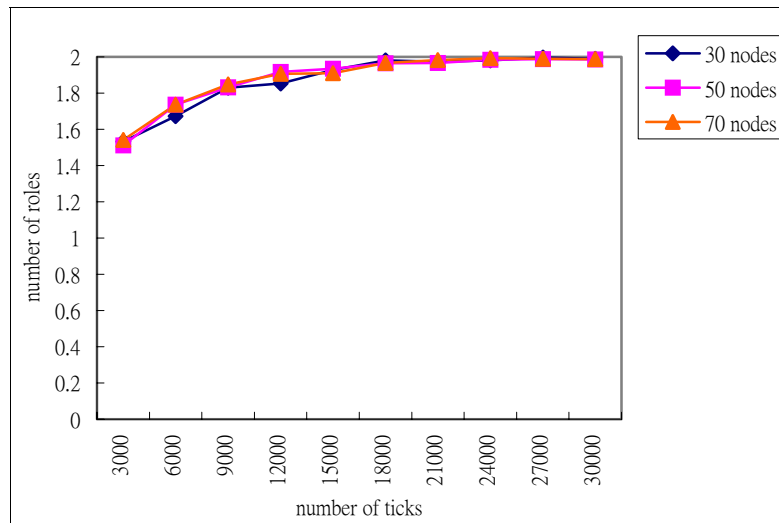


Fig. 12. Number of roles per node.

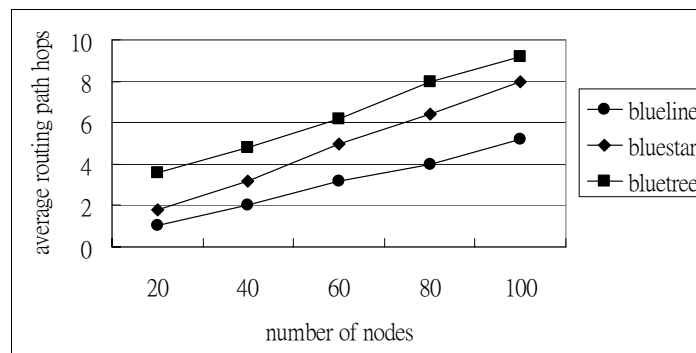


Fig. 13. Comparisons of average number of hops for routing paths.

5. CONCLUSIONS

In this paper, we have presented a solution to establish a multihop wireless ad hoc network based on Bluetooth technology and we also describe its routing algorithm. In Bluetooth specifications, scatternet formation algorithm should be implemented on the software level. Our algorithm is completely compatible with current Bluetooth specification. It can help current Bluetooth devices building scatternet without changing any hardware module. Blueline allows devices to self-organize themselves into piconets and finally form a single connected scatternet (provided that network connectivity is physically achievable).

We also analyze the effect of topology discovery performance on different probability of inquiry mode in each round by simulation. We find that a probability of 0.5 for being in inquiry scan mode in each round is not the best choice. It has better neighbor discovery performance when the probability is 0.75. This is an interesting phenomenon

and we are currently investigating its probable mathematical foundation. Since many slaves will be in the park mode according to our protocol, it is possible that the scatternet formed is more energy-efficient than others. This is also a possible future research topic. Finally, the mobility support of our algorithm is not discussed yet. We can make our algorithm support mobility by turning the neighbor discovery time to infinity. However, it needs further study.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive comments that greatly improve the quality of this paper.

REFERENCES

1. S. Baatz, C. Bieschke, M. Frank, C. Kühn, P. Martini, and C. Scholz, "Building efficient bluetooth scatternet topologies from 1-factors," in *Proceedings of the IASTED International Conference on Wireless and Optical Communications (WOC 2002)*, 2002, pp. 300-305.
2. C. J. Hsu and Y. J. Joung, "An ns-based bluetooth topology construction simulation environment," in *Proceedings of 36th Annual Simulation Symposium*, 2003, pp. 145-153.
3. C. J. Hsu and Y. J. Joung, *BlueHoc Extended Packages for Bluetooth Topology Construction*, http://joung.im.ntu.edu.tw/bluehoc_ex/.
4. A. Kumar, BlueHoc Manual; <http://oss.software.ibm.com/bluehoc/tutorial/docpage.html>, IBM India Research Lab.
5. C. Law, A. K. Mehta, and K. Y. Siu, "Bluetooth: performance of a new bluetooth scatternet formation protocol," in *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2001, pp. 183-192.
6. D. G. Leeper, "A long-term view of short and range wireless," *IEEE Computer Magazine*, Vol. 34, 2001, pp. 39-44.
7. T. Y. Lin, Y. C. Tseng, and K. M. Chang, "Formation, routing, and maintenance protocols for the bluering scatternet of bluetooths," *Wireless Communications and Mobile Computing*, Vol. 3, 2003, pp. 517-537.
8. Y. Liu, M. J. Lee, and T. N. Saadawi, "A Bluetooth scatternet-route structure for multihop ad hoc networks," *IEEE Journal on Selected Areas in Communications*, Vol. 21, 2003, pp. 229-239.
9. C. Petrioli, S. Basagni, and M. Chlamtac, "Configuring BlueStars: multihop scatternet formation for bluetooth networks," *IEEE Transactions on Computers*, Vol. 52, 2003, pp. 779-790.
10. T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire, "Distributed topology construction of bluetooth personal area networks," in *Proceeding of 20th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2001*, Vol. 3, 2001, pp. 1577-1586.
11. T. Sato and K. Mase, "A scatternet operation protocol for bluetooth ad hoc net-

- works,” in *Proceeding of 5th International Symposium on Wireless Personal Multimedia Communications*, Vol. 1, 2002, pp. 223-227.
12. M. T. Sun, C. K. Chang, and T. H. Lai, “A self-routing topology for bluetooth scatternets,” in *Proceeding of International Symposium on Parallel Architectures, Algorithms and Networks (I-SPAN '02)*, 2002, pp. 13-18.
 13. G. Tan, A. Miu, J. Guttag, and H. Balakrishnan, “Forming scatternets from bluetooth personal area networks,” MIT Technical Report, MIT-LCS-TR-826, Cambridge, MA, 2001.
 14. G. V. Zaruba, S. Basagni, and I. Chlamtac, “Bluetrees-scatternet formation to enable Bluetooth-based ad hoc networks,” in *Proceedings of IEEE International Conference on Communications*, Vol. 1, 2001, pp. 273-277.
 15. <http://www.bluetooth.com>, *Specification of the Bluetooth System*, Vol. 1, Core. Version 1.1, 2001.
 16. <http://www.isi.edu/nsnam/ns/>, The network simulator: NS-2.



Ruay-Shiung Chang (張瑞雄) received his B.S. degree from National Taiwan University in 1980 and his Ph.D. degree in Computer Science from National Tsing Hua University in 1988. He is now a Professor in the Department of Computer Science and Information Engineering, National Dong Hwa University. His research interests include Internet, wireless networks, and grid computing. Dr. Chang is a member of ACM and IEICE, a senior member of IEEE, and founding member of ROC Institute of Information and Computing Machinery. Dr. Chang also serves on the advisory council for the Public Interest Registry (www.pir.org).



Ming-Te Chou (周明德) received his B.S. degree from National Central University in 2001 and his M.S. degree in Computer Science from National Dong Hwa University in 2003. He is now a software research engineer in the Accton Corporation. His research interests include Internet, wireless networks, and multimedia.