A Bluetooth Scatternet Formation Algorithm for Networks with Heterogeneous Device Capabilities

Dimitri Reading-Picopoulos * and Alhussein A. Abouzeid

Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institute,
Troy, New York 12180-3590
ug99058@ee.ucl.ac.uk, abouzeid@ecse.rpi.edu

Abstract. This paper focuses on Bluetooth, a promising new wireless technology, developed mainly as a cable replacement. We argue that, in practice, Bluetooth devices will have different power capabilities, classifying them as either high-power or low-power nodes. We propose a deterministic, distributed algorithm that accounts for the physical properties of devices, connecting nodes into a scatternet of small diameter. The proposed protocol results in a high effective throughput and allows components to arrive and leave arbitrarily, dynamically updating the cluster formation. Performance is evaluated through extensive *ns-2* simulations.

1 Introduction and Problem Statement

Bluetooth [1–3] is rapidly emerging as the leading technology in the formation of short-range wireless *ad hoc* networks. The standard provides for low power wireless communication and operates in the 2.4GHz Industrial, Scientific and Medical band. Bluetooth connections are based on a master-slave configuration and employ a frequency hopping spread spectrum.

Bluetooth devices connect into *piconets*, each consisting of a master and up to seven slaves, while master-slave communication is achieved through time division duplexing. The Bluetooth specification [1] facilitates the connection of piconets into *scatternets* through common nodes. These devices can either participate in both piconets as slaves (*bridge device*) or as a master in one and as a slave in the other. The specification, however, does not allude to a specific mechanism by which scatternet formation is to be achieved.

This paper presents a new scatternet formation protocol that takes into account the physical limitations of the devices themselves. The resulting clustering has several attractive features and proves the importance of an efficient clustering algorithm.

Miklós *et al.* [4] take a statistical approach in studying the relationship between scatternet design rules and performance parameters. Raman *et al.* [5] argue for extensive cross-layer optimizations in Bluetooth scatternets, while Salonidis

* This work was performed while on an exchange program from University College London, UK, at Rensselaer Polytechnic Institute, Troy, NY, USA.

et al. [6] discuss the issue of fast connection establishment between Bluetooth devices and propose a symmetric protocol.

Ramachandran *et al.* [7] present two scatternet formation algorithms, but, in contrast to our work, no constraint was imposed on the number of *roles* assumed by the devices. The solutions to efficient scatternet formation presented by [8–10] are limited by the lack of a mechanism for minimizing the number of piconets in the resulting scatternet. An asynchronous distributed protocol for scatternet construction is presented by Salonidis *et al.* [11], while another scatternet formation algorithm is suggested by Law *et al.* [12], where the scatternet is formed once a leader is elected.

All previous works mentioned above have been performed without taking into consideration the physical limitations of the host units. Not all devices have the required processing power or battery lifetime to sustain a large number of slaves. We propose an algorithm that accounts for such restraints.

Bluetooth link formation is a two-step process with devices having to go through the *inquiry* and *page* states prior to establishing a connection. The purpose of the inquiry procedure is for a master node to obtain the Bluetooth 48-bit MAC address (BD_ADDR) and native clock (CLKN) of devices, which lie within its communication range. Connections are subsequently established through the paging mechanism using the information acquired during the inquiry procedure. Device discovery is potentially a time consuming process, while paging delays are much smaller. The complete state transition sequence, leading to a master-slave connection, is illustrated in Figure 1.

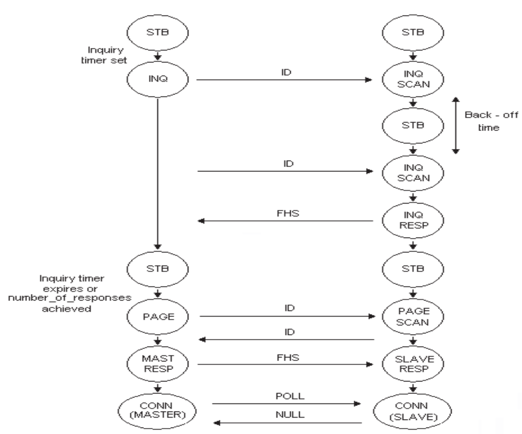

Fig. 1. State transitions leading to connection. STB is the STANDBY (idle) state. ID packets contain an inquiry access code, while Frequency Hop Synchronization (FHS) packets contain the information needed for frequency hop channel synchronization

The purpose of any distributed Bluetooth scatternet formation algorithm is the clustering of any group of asynchronous, isolated, Bluetooth devices to permit the communication of information between any pair of nodes. Such an algorithm should account for the following.

- Account for the nature of the host device itself, mainly its power limitations and battery lifetime.
- Restrict device participation to, at most, two piconets.
- Not all devices have to be within communication range of one another.
- Maximize the number of devices per piconet, thereby minimizing the number of piconets in the scatternet.
- The resulting network topology should be able to reconfigure, without causing long periods in the loss of connectivity, in the case of nodes arbitrarily joining or leaving the network.
- Path latency – the number of hops between any pair of devices – should be minimized.
- For simplicity, communication loops should be avoided.

The following section outlines our new scatternet formation algorithm. Section 3 introduces the simulation results, while section 4 concludes with some future research possibilities.

2 Scatternet Formation Algorithm

In this section, we present our Bluetooth scatternet formation algorithm based on the properties mentioned above. We differentiate between the following types of device.

high_Power: A host device with high processing power and adequate battery lifetime, capable of connecting, as master, with up to X nodes, of which up to Y are other high_Power nodes acting as slaves and the rest, W, are low_Power devices acting as slaves. Ideally, $Y = 0$ and $X = W$.

low_Power: Host peer with low processing power, ideally acting as slave. Only under extreme circumstances will such a device connect as master of a piconet, in which case it is limited to Z or less other low_Power nodes acting as its slaves. A special case of a low_Power device is a secure_Device.

secure_Device: A device that can only send and can neither receive nor forward information. Such a node can only connect as slave and then in a single cluster.

High_ and low_Power nodes assume one of the following roles when connected.

high_Power_Master: A high_Power peer that connects as master of a piconet.
low_Power_Slave: low_Power node that acts as slave in the resulting topology.
high_Power_Slave: high_Power device forced to assume the role of a slave.
low_Power_Master: low_Power node forced to act as master.
The algorithm is performed at every node and can be viewed as four stages.

2.1 Stage 1: Piconet Formation

Devices start off isolated, with no prior knowledge of the presence of other nodes. Stage 1 follows the Bluetooth specification [1], but is adapted to the existence of two types of device. By the end of the stage, the n nodes are partitioned into piconets, with a few devices potentially left unconnected. It is assumed that each device knows whether it is a high_ or a low_Power device.

A high_Power peer will enter Stage 1 of the algorithm in the INQUIRY state and wait for an INQUIRY RESPONSE for a period of T_{Inq} seconds. Upon reception of the FHS packet, it will subsequently enter the PAGE state and form the connection with the scanning device as its slave. This cycle is repeated until either T_{Inq} expires or the specified number of responses is achieved.

```
PICONET FORMATION
if (node is a high_Power device) then
  if((number of high_Power_Slaves)<Y and ((number of high_Power_Slaves) +
   (number of low_Power_Slaves))<X) then
    while (forever) perform INQUIRY
      if (INQUIRY RESPONSE received) then
        go to PAGE state →CONNECTION, go to PICONET FORMATION
      if (T_Inq expires or num_Responses achieved) then go to SCATTERNET FORMATION
    endwhile
  else go to SCATTERNET FORMATION
endif
```

Each time the high_Power node enters the INQUIRY state, T_{Inq} gets reset. Devices assigned as low_Power peers enter Stage 1 in the INQUIRY SCAN sub-state. Upon reception of an ID packet, such a device enters the PAGE SCAN state and subsequently connects to the paging unit. In the special case where the node is a secure_Device, the unit will then defer to the COMMUNICATION stage; otherwise it will enter the SCATTERNET FORMATION stage of the protocol. In the case where its clock, $T_{Inq-Scan}$, times out, the device will jump to the SCATTERNET FORMATION stage with secure_Devices returning to the INQUIRY SCAN sub-state.

```
PICONET FORMATION
if (node is a low_Power device) then
  while (forever) perform INQUIRY SCAN
    if (INQUIRY packet received) then
      send INQUIRY RESPONSE, enter PAGE SCAN→CONNECTION
      if (node is a secure_Device) then go to COMMUNICATION
      else go to SCATTERNET FORMATION
    if (T_Inq-Scan expires) then
      if (node is a secure_Device) then go to PICONET FORMATION
      else go to SCATTERNET FORMATION
  endwhile
endif
```

2.2 Stage 2: Scatternet Formation

The aim of Stage 2 is the interconnection of the piconets and the remaining isolated devices into a tree topology that spans the entire n nodes. Following a recommendation put forward by Salonidis *et al.* [6, 11], devices enter the second stage alternating between the INQUIRY and INQUIRY SCAN sub-states. The amount of time that each unit remains in a particular sub-state dictates T_{I-IS}, the overall time that a node should spend in Stage 2 attempting to connect into a scatternet. A unit alternating between the two states will enter the PAGE procedure upon reception of an INQUIRY RESPONSE sent by a device performing INQUIRY SCAN. The two nodes will then connect with the master device returning to Stage 2. Likewise, upon reception of an ID packet, the node will check for the possibility of the creation of a communication loop. Detecting a loop forces the node to return to the beginning of the second stage without resetting T_{I-IS}, otherwise, an INQUIRY RESPONSE is sent. Upon connection, the unit will reset T_{I-IS} and return to Stage 2, unless it connects as a bridge device, in which case it defers to Stage 3. If T_{I-IS} times out, the device defers to the third stage of the protocol.

```
SCATTERNET FORMATION
if ((high_Power peer and (number of high_Power_Slaves)<Y and ((number
   of high_Power_Slaves) + (number of low_Power_Slaves))<X) or
   (low_Power peer and (number of low_Power_Slaves)<Z)) then
   alternate between INQUIRY and INQUIRY SCAN states
   if (INQUIRY RESPONSE received) then
      enter PAGE state →CONNECTION, go to SCATTERNET FORMATION
   if (INQUIRY packet received) then
      if (loop detected) then do not stop clock, return to SCATTERNET FORMATION
      else send INQUIRY RESPONSE, enter PAGE SCAN→CONNECTION
         if (device becomes bridge) then go to SCATTERNET REORGANIZATION
         else reset clock, go to SCATTERNET FORMATION
   if (clock expires) then go to SCATTERNET REORGANIZATION
else perform INQUIRY SCAN
   if (INQUIRY packet received) then
      if (loop detected) then do not stop clock, return to SCATTERNET FORMATION
      else send INQUIRY RESPONSE, enter PAGE SCAN→CONNECTION
         go to SCATTERNET REORGANIZATION
   if (clock expires) then go to SCATTERNET REORGANIZATION
endelse
```

Loop detection: We propose a mechanism, which makes use of a DIAC (*Dedicated Inquiry Access Code*), which permits only restricted classes of devices to be inquired upon, in the transmission of ID packets by a node in INQUIRY.

Upon the establishment of a master-slave connection, the slave device will form a table, listing the BD_ADDR of its master. If this slave subsequently forms a piconet acting as a master, it will broadcast its table to all of its slaves, which will also append their own master's BD_ADDR. Similarly, each device forming a piconet as a master will request and acquire the tables of each of its slaves combining them to form its own table with the addition of its own BD_ADDR. This procedure ensures that every node in the network holds information on all master devices that it can access. Device table entries are appropriately updated every time a new master enters or leaves the network.

Given the "global" view that each node has of the network, a device in the INQUIRY sub-state will transmit ID packets containing its master's BD_ADDR as a DIAC. A node in INQUIRY SCAN will capture the DIAC and compare it to the entries in its table. Only if a match is not found do the two devices proceed in forming the connection. If a node in INQUIRY does not have a master it uses its own BD_ADDR as the DIAC.

2.3 Stage 3: Scatternet Reorganization

Stage 2 results in a tree topology that covers the entire network of devices and ensures connectivity between any pair of nodes. The resulting clustering, however, is not optimized in terms of minimum number of piconets and minimum path latency, as discussed earlier. Stage 3 seeks to reorganize the clustering in order to maximize network performance. We denote two master devices as x and y. These two units will either be connected directly, with one configured as a slave, or via a third, bridge, device. If both masters are either high_Power_Masters, or both are low_Power_Masters, the master with the highest number of slaves assumes the role of x. In the case where both peers have the same number of children, the master with the highest address becomes x. Similarly, in the situation where the two masters initiated the protocol as devices of opposite class, the master that started off as a low_Power node is configured as y. The low_Power_Slaves of x and y are represented as lp_S(x) and lp_S(y), while their high_Power_Slaves as hp_S(x) and hp_S(y) respectively. Each master device will run the procedure SCATTERNET REORGANIZATION for each master to whom it is directly connected and for every master separated through a bridge.

```
SCATTERNET REORGANIZATION
if (master a high_Power_Master or a low_Power_Master or a
   (single device and a high_Power peer)) then
   if (master has no one hop connection) then go to PICONET FORMATION
   if (one hop devices are not masters nor slaves in another piconet) then
      go to COMMUNICATION
   else go to COMBINE OR TRANSFER
else   if (one hop device is a master) then
          wait for scatternet reorganization, go to COMMUNICATION
       else go to PICONET FORMATION
endelse
```

The COMBINE OR TRANSFER subroutine performs the actual topology reformation and differentiates between the situation where two piconets can be merged into a single cluster and the case where devices can be transferred from one piconet to the other.

```
COMBINE OR TRANSFER
if (x, y are both low_Power_Masters) then
   if ((|lp_S(x)| + |lp_S(y)|)≤Z) then go to COMBINE.A
   else go to TRANSFER.A
else  if ((|lp_S(x)| + |hp_S(x)| + |lp_S(y)| + |hp_S(y)|)≤X) then go to COMBINE.B
      else go to TRANSFER.B
endelse
```

The master y will *convey* a slave device A by terminating their connection, with A entering the PAGE SCAN mode prior to connecting as a slave to master x. If the slave A cannot become a slave of master x (e.g. because it is out of its communication range or because A does not actually exist) then A is deemed *unavailable*. The procedure COMBINE.A is only run in the case where both x and y are low_Power_Masters with a combined number of slaves $\leq$Z.

```
COMBINE.A
if (low_Power_Slave of y available) then
   convey low_Power_Slave from y to x, go to COMBINE.A
else  if (x and y connected via bridge device and y available) then
         y disconnects from bridge, y connects to x as slave, go to COMMUNICATION
      else go to COMMUNICATION
endelse
```

For y and x both being low_Power_Masters with a joint sum of low_Power_Slaves >Z, a number of slaves will be *conveyed* from y to x with the procedure terminating once x obtains Z low_Power_Slaves.

```
TRANSFER.A
if (|lp_S(x)|< Z) then
   if (x and y connected via bridge device and y available) then
      y disconnects from bridge, y connects to x as slave, go to TRANSFER.A
   else  if (low_Power_Slave of y available) then
```

```
        convey low_Power_Slave from y to x, go to TRANSFER.A
      else go to COMMUNICATION
else go to COMMUNICATION
endelse
```

In the case where at least one of the two masters is a high_Power_Master, the equivalent procedures to COMBINE.A and TRANSFER.A are the following.

```
COMBINE.B
if (low_Power_Slave of y available) then
  convey low_Power_Slave from y to x, go to COMBINE.B
else  if (high_Power_Slave of y available and |hp_S(x)|< Y) then
        convey high_Power_Slave from y to x, go to COMBINE.B
      else  if (x and y connected via bridge device and y available) then
              if ((y is a high_Power_Master and |hp_S(x)|< Y) or
                (y is a low_Power_Master)) then
                y disconnects from bridge, y connects to x as slave, go to COMMUNICATION
              else go to COMMUNICATION
            else go to COMMUNICATION
endelse
```

```
TRANSFER.B
if ((|lp_S(x)| + |hp_S(x)|)<X) then
  if (x and y connected via bridge device and y available) then
    y disconnects from bridge, y connects to x as slave, go to TRANSFER.B
  else  if (low_Power_Slave of y available) then convey low_Power_Slave from y to x
          if ((|lp_S(x)| + |hp_S(x)|)<X) then
            if (high_Power_Slave of y available) then
              if (|hp_S(x)|<Y) then convey high_Power_Slave from y to x, go to TRANSFER.B
              else   if (low_Power_Slave of y unavailable) then go to COMMUNICATION
                     else go to TRANSFER.B
            else go to TRANSFER.B
          else go to COMMUNICATION
        else  if (high_Power_Slave of y available and |hp_S(x)|<Y) then
                convey high_Power_Slave from y to x, go to TRANSFER.B
              else go to COMMUNICATION
else go to COMMUNICATION
endelse
```

Efficient clustering can be achieved through scatternet reorganization provided the following rules are satisfied.

• In order to avoid an excessively high degree of inter-piconet overhead, either one of the two masters, x or y, has to run Stage 3 of the protocol with the complementary unit in COMMUNICATION.

• Piconets must be capable of updating dynamically. When two piconets are reconfigured through Stage 3, all participating devices will be instructed to inform their other masters (if they have any) to enter the third stage.

2.4 Stage 4: Communication

Devices in the COMMUNICATION stage spend their time between communicating information and trying to improve the overall connectivity of the network by periodically returning to Stage 2. The time spent by master devices between transitions, is made inversely proportional to their number of slaves. Equivalently, the inter-procedure switching of slaves is a function of the amount of traffic handled by the node, since idle units will be more willing to connect into new piconets. By permitting devices to return to earlier stages of the protocol, network healing and device assimilation can easily be accounted for.

3 Simulation Results

In this section we evaluate the performance characteristics of our protocol using *Bluehoc*, a Bluetooth performance evaluation tool developed by IBM [13], which provides a Bluetooth extension for *Network Simulator* [14]. The timer values defined in the first stage of our algorithm follow the recommended values given in the Generic Access Profile (GAP) of the Bluetooth specification [1]. When devices alternate between INQUIRY and INQUIRY SCAN, the mean, per state, residence time T_{Res}, based on the results obtained by Salonidis *et al.* [6], was chosen at $T_{Res} = 600msec$. Through simulations we found that a timeout period of 2.60 seconds for Stage 2 would complement the above value of T_{Res}.

The Bluetooth specification restricts $X = 7$. Using typical power consumption specifications we limit $Y = 4$ and $Z = 3$. In all simulations, Bluetooth units arrive uniformly over a *10 second* window and are placed randomly over a given geographical area. Not all devices are within communication range of each other and nodes are assumed to be static or of low mobility for the duration of the protocol. Results are plotted as the average of 10 simulation runs.

Scatternet formation time is dictated by the ratio of low_ to high_Power nodes in a particular geographical region. Devices configured as low_Power spend roughly three times more time, compared to high_Power peers, running the first stage of the algorithm in order to increase the probability of being discovered by an inquiring device. Figure 2 shows the average time taken for a node to establish its first connection, for a static number of 10 units, as a function of increasing number of high_Power nodes. As expected, connection time is maximized when all 10 devices are low_Power and is minimized when 40% of all nodes are high_Power. It is obvious that making the number of high_Power nodes adaptive to the network population can result in a low first connection setup time. In the rest of this section we will refer to the 3 : 2 ratio of low_ to high_Power peers as the *best case ratio* and the *worst case ratio*, the situation where all devices are low_Power.

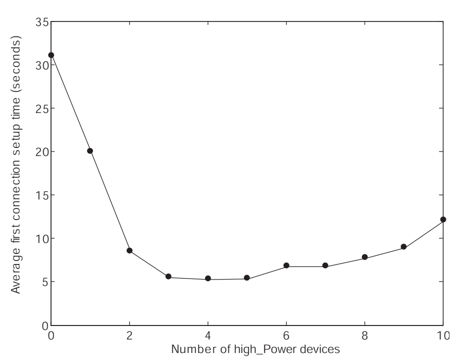

Fig. 2. Establishing the *best* and *worst case ratios*

Figure 3 illustrates the average time taken for scatternet formation, including reorganization, as a function of the number of devices within consideration. Both case ratios demonstrate an increase in topology construction time as the network size increases. For a given number of nodes, the *worst case ratio* results in a clustering time of roughly twice the duration compared to the time under the *best case ratio.*

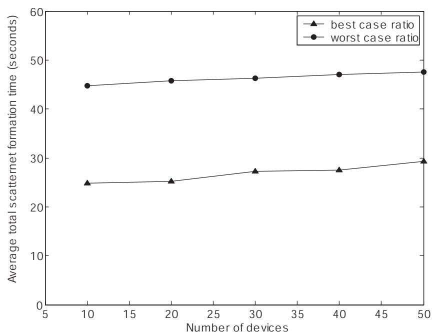

Fig. 3. Time required for scatternet formation, including reconfiguration time. Note that the devices arrive randomly within a 10 second time window, also included in the above results

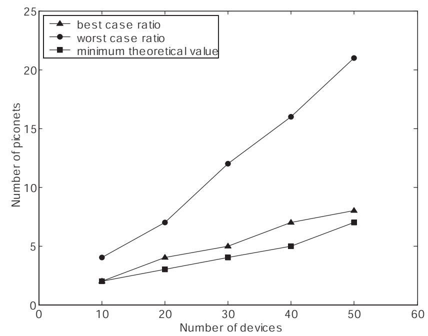

Fig. 4. Number of piconets in the resulting scatternets compared to the minimum theoretical value

Finally, Figure 4 compares the theoretical minimum number of piconets, with the average number of piconets for the *worst* and *best case ratios.* It can be easily shown that the theoretical minimum number of piconets is given by $max(\frac{n}{X+1}, \frac{NHP}{Y+1})$, where NHP is the number of high_Power nodes. Inter-piconet interference arises from adjacent piconets sharing the same frequency hopping sequence and results in a high degree of packet loss through repetitive collisions. Minimizing the number of piconets per scatternet serves to reduce this deterioration in network performance.

4 Conclusion and Future Work

In this paper we present and describe a new scatternet formation protocol. Even though the emphasis is put on the Bluetooth technology, the proposed algorithm is abstract enough as to provide for proficient clustering in wireless networks where link establishment is based on a master-slave relationship. Devices are divided into two categories depending on their power characteristics or other criteria and start off isolated with no prior knowledge of the presence of other nodes within their surroundings. The algorithm connects devices into a single scatternet, while attempting to minimize the number of piconets in the process.

No range constraint is imposed and the resulting topology is dynamically updateable to provide for devices arbitrarily leaving or wishing to join the network.

The simulation results show that making the number of high_Power devices adaptive to the network population greatly reduces scatternet connection delays. Network performance and resource utilization are maximized in the ideal situation of a 3 : 2 ratio between low_ and high_Power nodes and minimized in the case of networks containing 100% low_Power devices. Our future work includes quantitatively assessing the power savings obtained through the implementation of our algorithm and extending the proposed protocol to account for mobility.

References

1. Specification of the bluetooth system. http://www.bluetooth.com/ [October 2001].
2. J. Bray and C.F. Sturman. *Bluetooth Connect Without Cables*. Prentice Hall, 2001.
3. J.C. Haartsen and S. Mattisson. Bluetooth - a new low-power radio interface providing short-range connectivity. In *Proceedings of the IEEE*, volume 88, pages 1651–1661, October 2000.
4. G. Miklós, A. Rácz, Z. Turányi, A. Valkó, and P. Johansson. Performance aspects of bluetooth scatternet formation. In *Proceedings of The First Annual Workshop on Mobile Ad Hoc Networking and Computing*, pages 147–148, 2000.
5. B. Raman, P. Bhagwat, and S. Seshan. Arguments for cross-layer optimizations in bluetooth scatternets. In *Proceedings of Symposium on Applications and the Internet*, pages 176–84, 2001.
6. T. Salonidis, P. Bhagwat, and L. Tassiulas. Proximity awareness and fast connection establishment in bluetooth. In *First Annual Workshop on Mobile and Ad Hoc Networking and Computing*, pages 141–142, 2000.
7. L. Ramachandran, M. Kapoor, A. Sarkar, and A. Aggarwal. Clustering algorithms for wireless ad hoc networks. In *Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 54–63, Boston, MA, USA, August 2000.
8. G. Tan, A. Miu, J. Guttag, and H. Balakrishnan. Forming scatternets from bluetooth personal area networks. Technical Report MIT-LCS-TR-826, MIT, 2001.
9. Z. Wang, R.J. Thomas, and Z. Haas. Bluenet - a new scatternet formation scheme. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS-35)*, Big Island, HI, USA, January 2002.
10. G.V. Záruba, S. Basagni, and I. Chlamtac. Bluetrees - scatternet formation to enable bluetooth-based ad hoc networks. In *Proceedings of the IEEE International Conference on Communications*, volume 1, pages 273–277, Helsinki, Finland, 2001.
11. T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire. Distributed topology construction of bluetooth personal area networks. In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies*, 2001.
12. C. Law, A.K. Mehta, and K.-Y. Siu. Performance of a new bluetooth scatternet formation protocol. In *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing 2001*, Long Beach, California, USA, October 2001.
13. Bluehoc: Bluetooth ad-hoc network simulator. http://oss.software.ibm.com/developerworks/projects/bluehoc/ [October 2001].
14. The network simulator ns-2. http://www.isi.edu/nsnam/ns/ [October 2001].