Distributed Topology Construction of Bluetooth Personal Area Networks

Theodoros Salonidis[1], Pravin Bhagwat[2], Leandros Tassiulas[1], and Richard LaMaire[3]
thsalon@glue.umd.edu, pravinb@research.att.com, leandros@glue.umd.edu, lamaire@us.ibm.com

[1]Electrical and Computer Engineering Department, University of Maryland at College Park.
[2]AT&T-Labs Research, Florham Park NJ.
[3]IBM T.J. Watson Research Center, Hawthorne NY.

Abstract-- In recent years, wireless ad hoc networks have been a growing area of research. While there has been considerable research on the topic of routing in such networks, the topic of topology creation has not received due attention. This is because almost all ad hoc networks to date have been built on top of a single channel, broadcast based wireless media, such as 802.11 or IR LANs. For such networks the distance relationship between the nodes implicitly (and uniquely) determines the topology of the ad hoc network.

Bluetooth is a promising new wireless technology, which enables portable devices to form short-range wireless ad hoc networks and is based on a frequency hopping physical layer. This fact implies that hosts are not able to communicate unless they have previously discovered each other by synchronizing their frequency hopping patterns. Thus, even if all nodes are within direct communication range of each other, only those nodes which are synchronized with the transmitter can hear the transmission. To support any-to-any communication, nodes must be synchronized so that the pairs of nodes (which can communicate with each other) together form a connected graph.

Using Bluetooth as an example, this paper first provides deeper insights into the issue to link establishment in frequency hopping wireless systems. It then introduces the Bluetooth Topology Costruction Protocol (BTCP), an asynchronous distributed protocol for constructing scatternets which starts with nodes that have no knowledge of their surroundings and terminates with the formation of a connected network satisfying all connectivity constraints posed by the Bluetooth technology. To the best of our knowledge, the work presented in this paper is the first attempt at building Bluetooth scatternets using distributed logic and is quite "practical" in the sense that it can be implemented using the communication primitives offered by the Bluetooth 1.0 specifications.

***Index terms*—Frequency hopping, Bluetooth, topology construction, scatternet.**

1. INTRODUCTION

An ad hoc network is a wireless network formed by nodes that cooperate with each other to forward packets in the network. Almost all experimental ad hoc networks to date have been built on top of single channel, broadcast based 802.11 wireless LANs or IR LANs. In such networks, all nodes within direct communication range of each other share a common channel using a CSMA style MAC protocol. In addition, multi-hop routing is used as a means for forwarding packets beyond the communication range of the source's transmitter. Since a single channel is used throughout the network, the topology of the ad hoc network is implicitly (and uniquely) determined by distance relationship among the participating nodes.

This paper is aimed at addressing a new problem which arises when multiple channels are available for communication in an ad hoc network. The problem is that of determining which subgroup of nodes should share a common channel and which nodes should act as relays and forward traffic from one channel to another. The channel assignment should be done so that all constraints posed by the underlying physical layer are satisfied while ensuring that the resultant graph formed by all nodes is connected.

We address an instance of the above problem which occurs in Bluetooth based ad hoc networks, known as scatternets [9]. Bluetooth is a promising new technology which is aimed at supporting wireless connectivity among cell phones, headsets, PDAs, digital cameras, and laptop computers. Initially, the technology will be used as a replacement for cables, but in due course of time solutions for point-to-multipoint and multi-hop networking over Bluetooth will evolve.

Bluetooth is a frequency hopping system which defines multiple channels for communication (each channel defined by a different frequency hopping sequence). A group of devices sharing a common channel is called a piconet. Each piconet has a master unit which selects a frequency hopping sequence for the piconet and controls the access to the channel. Other participants of the group known as slave units are synchronized to the hopping sequence of the piconet master. Within a piconet, the channel is shared using a slotted time division duplex (TDD) protocol where a master uses a polling style protocol to allocate time-slots to slave nodes. The maximum number of slaves that can simultaneously be active in a piconet is seven.

Multiple piconets can co-exist in a common area because each piconet uses a different hopping sequence. Piconets can also be interconnected via bridge nodes to form a bigger ad hoc network known as a scatternet. Bridge nodes are capable of timesharing between multiple piconets, receiving data from one piconet and forwarding it to another. There is no restriction on the role a bridge node can play in each piconet it participates in. A bridge can be a master in one piconet and slave in another (termed as M/S bridge) or a slave in all piconets (termed as S/S bridge).

This work is supported by an IBM University Partnership Award.

It is possible to organize a given set of Bluetooth devices in many different configurations. Figures 1b and 1c show two example configurations in which nodes in a Bluetooth network can be arranged. All nodes are assumed to be in radio proximity of each other. Fig. 1b shows an example in which all nodes are part of a single piconet[1]. Figure 1c illustrates another configuration in which node A is master of piconet 1, node E is master of piconet 3, node B is an M/S bridge (master of piconet 2 and a slave of piconet 1), node D is a slave of piconet 1 and node C is an S/S bridge (slave in piconets 2 and 3). In contrast to the above two configurations the node interconnection topology in a single channel system will be a complete graph (Fig. 1a) since all nodes will hear each other's transmission.

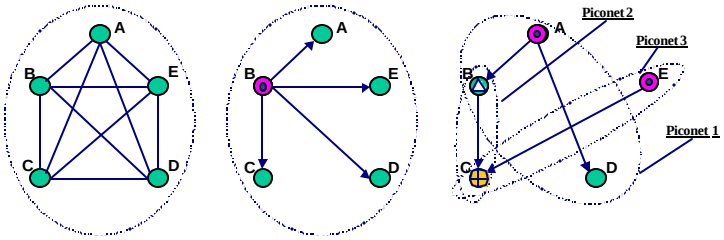

Figure 1: (a) Single channel model. (b),(c) Different configurations according to the Bluetooth multiple channel model.

Given a collection of Bluetooth devices, an explicit topology construction protocol is needed for forming piconets, assigning slaves to piconets, and interconnecting them via bridges such that the resulting scatternet is connected. Such a protocol should be asynchronous, totally distributed and nodes should start with no information about their surroundings. The problem of constructing distributed self-organizing networks has been addressed in the past ([4][5][6][10]), but all the efforts so far were aimed at solving the problem by assuming a single broadcast channel and a CSMA style MAC protocol. The problem is significantly harder for frequency hopping based wireless systems as will be evident in the later discussion.

This paper is a first attempt to address the topology construction problem in the multiple FH channel setting imposed by the Bluetooth technology. In order to solve it, we design our protocol in a bottom-up fashion: First, in section 2 we examine the wireless link provided by Bluetooth by presenting the asymmetric "sender-receiver" point to point link establishment protocol as defined in the Bluetooth specifications. In section 3 we enhance this protocol by proposing a symmetric variant of the link establishment protocol where two devices alternate independently between the "sender" and "receiver" state until they discover and connect to each other. Such a protocol is necessary for establishing a connection between a pair of identical devices or in situations when any external means for selecting initial device states are not available. Section 4 introduces the Bluetooth Topology Construction Protocol (BTCP), which is an asynchronous distributed connection establishment protocol that extends the point to point symmetric protocol to the case of many nodes. This protocol is based on a leader election process where each node uses a timeout to independently decide about the leader election termination. The timeout delay factor introduces a correctness-delay tradeoff of the network formation. By using the delay analysis of section 3 we show in section 5 how to best choose the protocol parameters in order to maximize the probability of forming a connected scatternet while minimizing delays. Finally, section 6 provides a future work discussion and conclusions.

2. LINK ESTABLISHMENT IN BLUETOOTH : BACKGROUND

The Bluetooth Baseband Specification [1] defines the Bluetooth point to point connection establishment as a two-step procedure. First neighborhood information is collected through the Inquiry Procedure. The Paging procedure is subsequently used to establish the connections between neighboring devices. Both the Inquiry and Paging procedures are asymmetric processes; they involve two types of nodes (which we call senders and receivers) each performing different actions. During Inquiry, "senders" discover and collect neighborhood information provided by "receivers". During Paging, "senders" connect to "receivers" discovered during a previous inquiry procedure.

During the inquiry or paging procedure, although senders and receivers use the same (inquiry or paging) frequency hopping sequence[2], it is likely that they will be out of phase since each unit starts at a different hop frequency derived from its local clock value. This (unavoidable) phase difference introduces a **phase uncertainty** among the devices participating in the procedure. To overcome this phase uncertainty, senders and receivers hop at different speeds. A receiver hops at a slow rate over the common frequency pattern listening on each hop for sender messages and the sender transmits at a much higher rate listening in between transmissions for an answer, in hope of discovering the frequency a receiver is currently listening to. Given two units, one operating as a sender and the other as a receiver, the term **Frequency Synchronization delay** (or **FS delay**) refers to the time until the sender transmits at the frequency the receiver is currently listening on[3].

Even if the two procedures have the same synchronization mechanism, a difference is that during the paging procedure the sender tries to bypass the FS delay by estimating the phase of the receiver. If paging is performed directly after the

[1] Note there is no edge among slave nodes since slaves cannot hear each other's transmission.

[2] N_f, the number of frequencies in the inquiry or page hopping set, is equal to 32 for systems operating in Europe and US and 16 for systems operating in Japan, Spain and France.

[3] The sender can cover the entire inquiry hopping frequency set in time $T_{coverage} = N_f \times 625us$ which is 10ms (20ms) for the 16 (32) hop system.

Inquiry procedure, the sender has acquired the clock value of the receiver unit and can use it to determine its phase and connect to it instantaneously.

The functional difference between the Inquiry and Paging Procedures lies in the use of a universal FH sequence in the first and a common point to point FH sequence in the second. Using a universal inquiry hopping sequence, a sender node effectively "broadcasts" an Inquiry Access Code (IAC) packet that can be heard only by receiver nodes that listen for such a packet. During the paging procedure, by using the receiver's page hopping sequence a sender node initiates connection establishment by effectively "unicasting" a Device Access Code (DAC) packet that can be heard only by the corresponding receiver device. Thus the Inquiry Procedure involves many units, where a sender can discover more than one receivers while the paging procedure involves only two units, where a sender pages and connects to a specific receiver.

2.1. The Bluetooth Asymmetric protocol for link formation

According to the Bluetooth Baseband specification the protocol starts by the sender starting in the INQUIRY state and the receiver in the INQUIRY SCAN state. As was described in the previous section there is an initial FS delay until the sender hits the frequency the receiver is listening to. Upon receiving the IAC packet, the receiver backs off for an amount of time that is uniformly distributed between 0 and 639.375ms. This happens in order to prevent the contention problem that would arise if there were two receivers listening on the same hop frequency. If both of them responded immediately, the response message would get garbled and the sender would not receive it. We call the time while the receiver backs off the **Random Backoff delay (or RB delay).** When the receiver unit wakes up, it starts listening again at the hop it was listening to before backing off. After a second FS delay (same as the first one), a second IAC packet is received from the sender. Then the receiver sends back to the sender an FHS packet that contains:

1. The receiver's address: This is used by the sender to derive the DAC of the receiver and the page hopping sequence it will use later in order to page the receiver.
2. The receiver's clock value: This is used to estimate the phase of the receiver and thus eliminate the FS delay during the paging procedure that follows.

The timing diagram in Figure 2, summarizes the point to point connection establishment procedure between the two units. The dashed arrows denote events on each unit's timeline and each event is numbered in the order it happens during the connection establishment procedure. The timing diagram shows that the receiver enters the PAGE SCAN state after sending the inquiry response FHS packet to the sender. When the sender receives the FHS packet, it enters the PAGE state and uses the clock information in the FHS packet to send a DAC packet on the frequency the receiver is listening to in the PAGE SCAN state. Then the receiver responds immediately with a DAC packet and the sender sends an FHS packet to the receiver. The receiver uses the FHS information to determine the channel hopping sequence and the phase of the sender and becomes the slave of the point to point connection. It then acknowledges the FHS packet with another DAC packet. As soon as the sender receives the acknowledgment, it becomes the master of the connection and may start exchanging data with the synchronized receiver-slave.

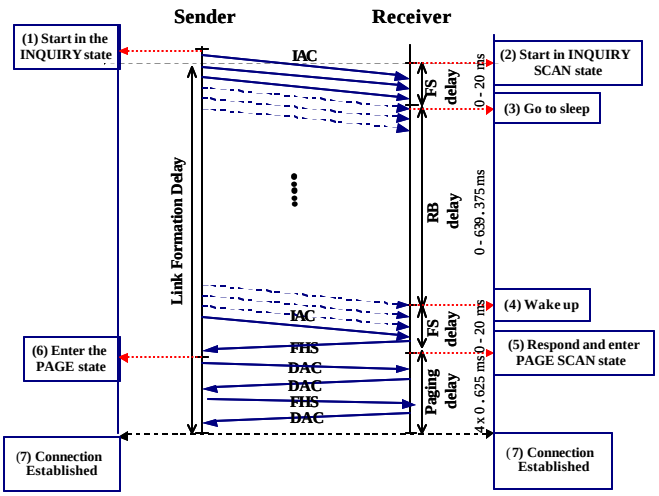

Figure 2: The Bluetooth asymmetric link formation protocol.

By observing Figure 2, we can easily identify the link formation delay components. The inquiry procedure delay consists of a first FS delay, the RB delay and a second FS delay that is taking place when the receiver waits for the second IAC packet after it wakes up. The paging procedure delay is negligible since it immediately follows the inquiry procedure. (As soon as the first DAC packet is received by the receiver the rest of the steps are happening in consecutive 625μs slots). Thus we can approximate the link formation delay *R* using the following equation:

$$R = 2FS + RB \qquad (1)$$

where *FS* and *RB* are uniform random variables in $[0, T_{coverage}]$ and $[0, r_{\max}]$ respectively. According to equation (1), the link formation delay can be at most $2T_{coverage} + r_{\max} = 20ms + 639.375ms = 659.375ms$ for the 32-hop system and *649.375ms* for the 16-hop system.

3. A SYMMETRIC PROTOCOL FOR LINK FORMATION

The asymmetric protocol provided by the Bluetooth specification, yields a very short connection establishment delay provided that the sender and receiver roles are pre-assigned.

When two or more users are trying to establish links between their Bluetooth devices in an ad hoc fashion, they will not be able to explicitly assign sender and receiver roles. They

will just press a button and expect to connect with their peers. Thus there should be a symmetric mechanism that forms connections in an ad hoc fashion without any explicit sender or receiver role pre-assignment. A way to do this, is by forcing the two nodes to alternate independently between the sender (INQUIRY state) and receiver (INQUIRY SCAN state) roles and try to connect according to the asymmetric protocol during an overlap interval where they meet in opposite states.

In Figure 3, Unit A has already started alternating, and Unit B starts alternating at some arbitrary time t_0. The merged schedule is produced by merging the state switching times of the two units into a single one, which can be seen as an "on-off" process.

By using state alteration, a connection will be established after a random delay, which in principle will be larger than the one of the asymmetric protocol. The reason is that starting at each "on" interval of the merged process, the two units will connect after a random interval $R = 2FS + RB$, given that they both remain fixed at their (complementary) states for an amount of time greater than *R*. Otherwise, they have to wait for the next "on" interval. The time T_c from time t_0 up to the point where the two units come to a complementary state for a sufficient amount of time is essentially the link formation delay between the two units.

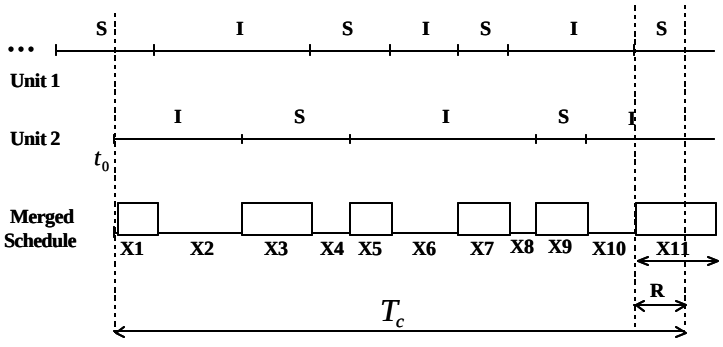

Figure 3: A symmetric link formation protocol: Nodes alternate between sender and receiver state until they connect.

There are some interesting questions arising from the proposed "alternating states" technique. First of all what should the alternating schedule be? Should the states alternate in a periodic or random fashion? It can be analytically proven that the mean connection time is infinite when each unit changes states deterministically. A formal proof is omitted due to lack of space and can be found in [3]. Intuitively, if the state residence intervals are fixed, the intervals of the merged process in Figure 3 will be fixed as well. Then the connection time will depend on the fixed phase difference of the two devices. If this phase is very small, then the "on" intervals in the merged process will be very small and the link formation delay very large since the units will use arbitrarily many "on" intervals until they finally connect.

Alternatively, a random schedule can be imposed on the state residence times. In [3] we provide an ad hoc link formation delay model, and show that when each unit alternates independently between INQUIRY and INQUIRY SCAN with the state residence times following a common random distribution, we can analytically calculate the mean and variance of the link formation delay.

The way to calculate the connection set up delay is to determine the cdf and pdf of the merged schedule process X given that the two nodes alternate independently according to an identical distribution Z. In [3] we show that the mean and variance of the link formation delay of the symmetric protocol are given by:

$$E[T_c] = \frac{E[X]}{2} + \frac{(E[X \mid R > X] + E[X])(1-p)}{p} + E[R] \qquad (2.1)$$

$$Var[T_c] = \frac{Var[X]}{2} + \frac{(Var[X \mid R > X] + Var[X])(1-p)}{p} + Var[R] \qquad (2.2)$$

where $p = P[R \le X]$. (2.3)

The "alternating states" technique is a mechanism that guarantees an ad hoc point to point connection between two Bluetooth devices. When more than two devices exist and wish to form a scatternet "on the fly", a protocol should be devised on top of this mechanism, that ensures that the resulting network will fulfill the requirements and structure of a Bluetooth scatternet. This protocol should also be efficient in terms of network establishment delay. We will use the symmetric link formation delay model derived in [3] in order to achieve this.

4. BTCP: A DISTRIBUTED SCATTERNET FORMATION PROTOCOL

Our motivation for the scatternet formation problem arises from a "conference-scenario" of an ad hoc network establishment. Suppose that there are many users in a room that wish to form an ad hoc network using their Bluetooth enabled devices. Each user presses a "start" button and waits for the device to show on the screen a "network connection established" message after a short period of time. After this message appears, the user will be able to exchange information with any other user in the room. The description of this application actually contains the elements of a successful connection establishment protocol:

- Network connection establishment should be performed in a totally distributed fashion. This means that each device starts operating asynchronously on its own and it initially does not have any knowledge about the identities or number of nodes in the room.
- After completion, the protocol must guarantee a connected scatternet. "Connected" means that there should be at least one path between any two nodes in the network.
- The network set up delay should be minimized such that it is tolerable by the end user.

In general there are no restrictions regarding the final form of the scatternet. The only requirements are that:

- There should be piconets that have one master and less than seven slaves and that piconets are interconnected through S/S or M/S bridge nodes.
- Every node must be able to reach every other node in the resulting network i.e. the network must be connected.

In addition to satisfying connectivity, a desirable feature of the protocol would be to be able to shape the network topology according to scatternet formation criteria imposed by specific applications. For example the same node may need to have different roles in different applications. Also it may be possible for a node to have more restrictive degree constraints than seven due to its own nature as a device; for example a palm pilot would not have the processing power to be a master of a seven slave piconet. Scatternet formation criteria could also be in the form of traffic demands that need to be satisfied by the nodes participating in the network construction process. These criteria should be taken into account during the topology construction process if they exist. The problem of defining scatternet forrmation criteria is itself an open research issue that is heavily dependent on the envisioned applications. Although we do not address it in this paper, our approach takes it into account by collecting information about all nodes participating in the process at a single point before actual connection happens.

BTCP is based on a leader election process. Leader election is generally an important tool for breaking symmetry in a distributed system. Since the nodes start asynchronously and without any knowledge of the total number of participating nodes in the network construction process, an elected coordinator will be able to control the network formation and ensure that the resulting topology will satisfy the connectivity requirements of a Bluetooth scatternet.

In the absence of any scatternet formation criteria, and in order to design a simpler and faster protocol, we propose and justify the following default properties that the resulting network will satisfy:

1. **A bridge node may connect only two piconets. (Bridge degree constraint):** A bridge node forwards data from one piconet to another by switching between them in a time division manner. Given that each portable device may have limited processing capabilities, a maximum bridge degree of two relieves a node of being an overloaded crossroad of multiply originated data transfers.
2. **Given the number of nodes N, the resulting scatternet should consist of the minimum number of piconets possible**. The impact of this is similar to the motivation of solving the problem in [6] of finding the minimum number of routers in an ad hoc network. A minimum number of piconets yields an easier network to control.
3. **The resulting scatternet should be fully connected**. This means that every master will be connected to all other masters through bridge nodes. Scatternets are expected to change and be reformed over time. A fully connected scatternet in its initial state provides higher robustness against topology changes. Also no routing is needed in this original state since every master can reach every other master through a bridge node and every slave can reach everybody else through its own master.
4. **Two piconets share only one bridge (Piconet overlap constraint).** This condition is used in order to provide a means of terminating easily the connection establishment protocol and calculating the minimum number of piconets. If two masters later wish to share another bridge between them they can do so by means of a bridge negotiation protocol.

The protocol consists of three phases:

Phase I: Coordinator Election

During this phase, there is an asynchronous, distributed election of a coordinator node that will eventually know the count, identities and clocks of all the nodes participating in the network construction process.

Each node x has a variable called VOTES which is set to 1 as soon as the node is powered up. After initialization, the node starts alternating between the INQUIRY and INQUIRY SCAN state.

Any two nodes x and y that discover each other will form a point to point connection, enter a “one-to-one confrontation” and compare their VOTES variables. The node with the larger variable is the winner of the confrontation. If the two nodes have equal VOTES variables the winner is the node with the larger Bluetooth address.

Without loss of generality, suppose that x is the winner and y is the loser. The loser y sends all the device FHS packets of the nodes it has won so far to the winner x, it tears down the connection and enters the PAGE SCAN state. In this way it will not be able to hear inquiry messages any more but only page messages from nodes that will page it in the future. This action has the effect of eliminating the loser from the coordinator election process and preparing it for the next phases of the protocol.

The winner x increases its VOTES variable by VOTES(y) and continues on the leader election process by resuming alternating between INQUIRY and INQUIRY SCAN.

If there are N nodes participating in the scatternet formation, there will be N-1 one-to-one confrontations. The winner of the N-1st confrontation will be the coordinator node and the rest of the nodes will be in the PAGE SCAN state waiting to be paged by a node that has information about them.

Phase II: Role Determination

The coordinator that was elected during phase I, has the FHS packets (i.e. identities+clocks) of all the nodes and hence knows the total number of nodes N that participate in the network connection establishment.

At the start of phase II, the coordinator checks if the number of nodes that it has discovered during phase I is less than eight. If this is the case, it pages and connects to all of the nodes in PAGE SCAN and one piconet is formed with the coordinator as the master and all the other nodes as its slaves. In this special case the protocol terminates at this point.

If the number of nodes is greater than seven then more than one piconet must be formed and interconnected via bridge nodes. Given the global view of the network the coordinator can decide on the role that each node will perform in the final scatternet. If the participating nodes impose specific scatternet formation criteria, they can be communicated to the coordinator during the election process in addition to the FHS information, and can aid it in determining the roles of the nodes in the final scatternet. By using the default criteria cited at the start of this section the coordinator first calculates the number of piconets P.

The minimum number of masters P in order for the resulting scatternet to be fully connected can be calculated by the following relation (see [3]):

$$P = \left\lceil \frac{17 - \sqrt{289 - 8N}}{2} \right\rceil, \quad 1 \le N \le 36 \qquad (3)$$

As we observe from the above relation, the default scheme works for a number of nodes less than or equal to 36 due to the desired properties 2-4 described at the beginning of this section. A larger number of nodes may lead to a default scheme that does not require a fully connected scatternet.

After calculating P, the coordinator selects itself and P-1 nodes to be the designated masters and $\frac{P(P-1)}{2}$ other nodes to be the scatternet bridges. Consequently, the coordinator equally distributes to the designated masters the remaining nodes to be their "pure" slaves.

After the role assignment, for each master x (including itself), the coordinator has a connectivity list set (SLAVESLIST(x), BRIDGELIST(x)) consisting of the master's assigned slaves and bridges. Each entry of these lists contains FHS packets (identities+clocks) so that the designated master can later page its connectivity list set instantaneously.

Then the coordinator connects to the designated masters it selected by paging them. (Recall that at the end of phase I all remaining nodes were in the PAGE SCAN state). Thus a temporary piconet is formed instantly with the coordinator as the "master" and the designated masters as the "slaves"[4]. The coordinator transmits to each designated master its connectivity list set, instructs the designated masters to start phase III, and consequently tears down the temporary piconet and starts phase III as a master node itself.

Phase III: The actual connection establishment

During this phase, each master x pages and connects to the slaves and bridges defined in its SLAVESLIST(x) and BRIDGELIST(x) respectively.

As soon as a node is notified by its master that it is a bridge, it waits to be paged by its second master. When this happens, the bridge node sends a CONNECTED notification to both masters.

When a master receives a CONNECTED notification from **all** its assigned bridges, a fully connected scatternet of P piconets is guaranteed to be formed and the protocol terminates.

It is evident that the most time consuming part of the protocol is the leader election phase. Phase II and Phase III involve only paging and connecting which is happening almost instantaneously due to the previous discovery phase. The tricky part of the protocol is actually the phase I termination. Ideally it should stop as soon as the coordinator is found. But how does a node know that it is the final winner of the election process? All nodes have a "state alternation" timeout period ALT_TIMEOUT that is set once a node is powered up and reset each time it wins an "one to one confrontation".

When ALT_TIMEOUT expires, the node assumes it is the elected coordinator and that all other nodes are in the PAGE SCAN state waiting to be paged.

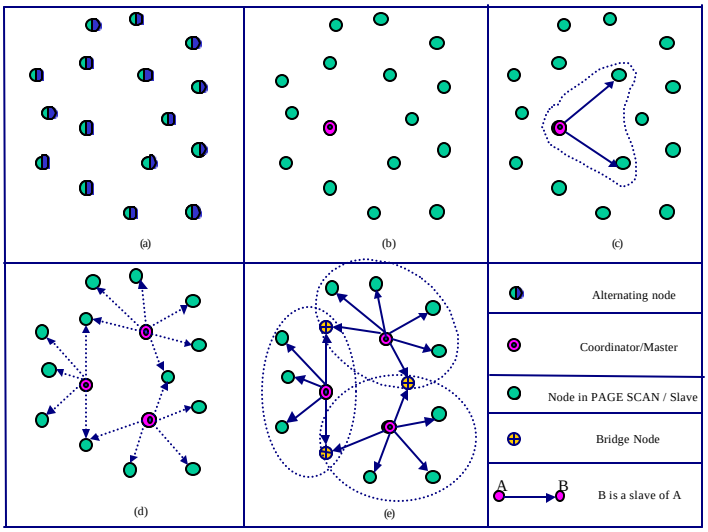

Figure 4: The connection establishment protocol for a set of N=16 nodes. (a) Start of Phase I: All nodes start alternating trying to discover their neighborhood. (b) At the end of phase I the coordinator has been elected. Since N=16 the coordinator computes P=3 and selects the masters, bridges and slaves accordingly (c) Phase II: Coordinator forms a temporary piconet with the designated masters and sends them their connectivity lists. (d) Phase III: Each master pages the nodes specified within its connectivity list. (e) Final scatternet formation.

[4] Note that according to equation (3), P is always less than seven. Thus the temporary piconet can always be formed.

The question that is raised now is "what is a good value for ALT_TIMEOUT"? A very large value will result in a node having won the competition and continuing alternating without knowing it is the only one left. This will result in a very slow phase I (and hence a very slow connection establishment protocol). On the other hand a very small timeout value may result in a case where more than one nodes assume they are the coordinator and hence a protocol that will result in a disconnected scatternet.

We address the above problem by making the following observation. When there are N nodes alternating and trying to discover and connect to each other, the time for the first connection to happen is generally less than the time it takes if there were only two of them trying to connect. According to the link model in [3], given a distribution and the mean state residence time, the mean connection establishment time can be analytically calculated for the two-node case and this value can be used to determine the ALT_TIMEOUT timer of each node.

5. EXPERIMENTS

5.1. Emulating Bluetooth

We have implemented BTCP on top of an existing prototype implementation that emulates the Bluetooth environment on a Linux platform. The reason for using an emulator instead of the Bluetooth devices themselves is because current Bluetooth units do not support the piconet switching function and hence cannot operate as bridges. In addition, an emulator provides a higher degree of flexibility in testing the system for various parameters and can afford testing the protocol for a large number of nodes.

Each Bluetooth host is implemented as a process that mainly consists of two interacting modules. The Bluetooth Baseband (BB) module emulates in software the Inquiry, Paging and piconet switching procedures as defined in the Bluetooth Baseband specification [1]. The BTCP module interacts with the BB module through the HCI control specification functions as defined in [2]. The use of HCI functions allows us to later replace the Bluetooth software module with a hardware module, when the bridging capabilities become available in hardware.

The wireless medium is simulated by a N_f-hop channel process which is used for the exchange of IAC and FHS packets during the inquiry and paging procedures. The N_f-hop channel process also determines the frequency hopping collisions that are happening between the devices and emulates the FS delays. Note that this channel process is not similar to a CSMA channel since the senders or receivers cannot perform carrier sensing or any kind of intelligent backoff.

We also assume that all the devices are within range of each other. This is a logical assumption for networking many short-range wireless devices in a single room. This fact is mapped in our architecture by having all Bluetooth host processes initially connected to the N_f-hop wireless channel process and executing the topology construction protocol.

5.2. Determining ALT_TIMEOUT

Using the the Periodic_Inquiry_Mode HCI command [2], it is possible to program Bluetooth units to alternate between INQUIRY and INQUIRY SCAN states with uniformly distributed state residence times. In this case the cdf of the merged process X (see Figure 3) when each unit has state residence times uniformly distributed in [0,b] is:

$$F_x(x) = P[X \le x] = \frac{1}{b^3}\cdot x^3 - \frac{3}{b^2}\cdot x^2 + \frac{3}{b}\cdot x, \qquad 0 \le x \le b \quad (4)$$

by using (4) and (1) in (2.1),(2.2),(2.3) we can calculate analytical expressions for the mean and variance[5] of the link formation time T_c of the symmetric protocol.

Given (2.1) and (2.2), we choose ALT_TIMEOUT according to the empirical relation:

$$ALT\_TIMEOUT = E[T_c] + \sqrt{Var[T_c]} + r_{\max} \quad (5)$$

Figure 5 shows the mean connection establishment time and the standard deviation in the connection establishment time in the point to point case when the state residence times are uniformly distributed. We observe that for every alternating mean state residence time the resulting standard deviation is almost equal to the mean connection time. This means that the link formation time distribution is not centered around the mean. This justifies the inclusion of the $\sqrt{Var(T_c)}$ term in our empirical formula.

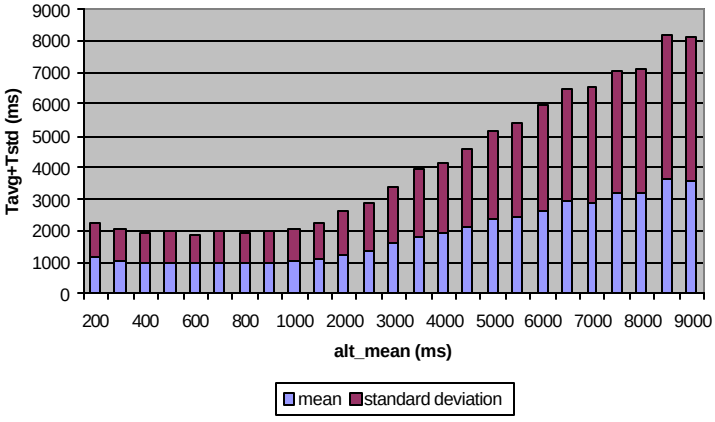

Figure 5: Means and standard deviation delays for the point to point connection establishment time where nodes alternate with state residence times according to a uniform distribution.

The term r_{max} was more subtle and was determined only after performing experiments and observing the protocol behavior on many runs. It seems like the following case was happening

[5] The analytical expressions can be found in [3].

very frequently: After the N-2nd confrontation the winner A would start alternating by resetting ALT_TIMEOUT while there was one node B in SLEEP mode (and all the rest in PAGE SCAN). The two nodes A and B would start trying to form the N-1st connection only after node B woke up! The additional term r_{max} is the upper limit for the backoff interval and thus eliminates the concern about this case.

In our experiments we choose a mean state residence time of 600ms which according to equation (5) and Figure 5 yields the smallest ALT_TIMEOUT value of 2527.223ms

5.3. Protocol Performance

The performance metrics associated with the protocol are the network connection set up delay and the probability of protocol correctness which depends on the value of ALT_TIMEOUT. The higher this value is, the higher the probability of protocol correctness but also the longer it will take the network to connect.

The network connection set up delay measured in the experiments is always the time to elect the leader (phase I duration) since phase II and III include only instantaneous paging and connections.

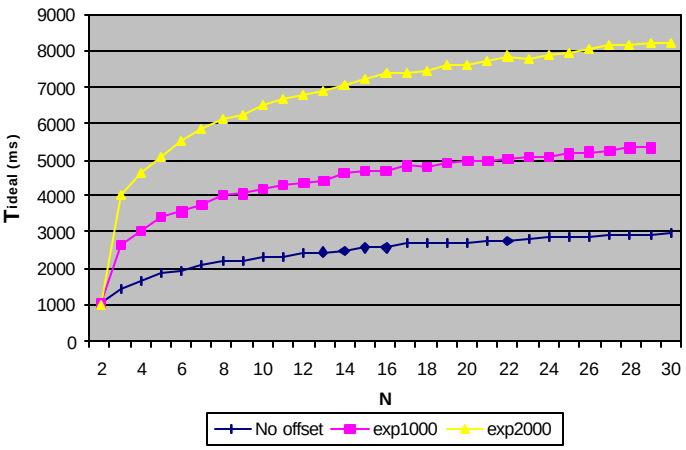

Figure 6: Average ideal connection establishment time for various application scenarios. Units alternate according to uniformly distributed state residence times with mean 1000ms.

The "no offset" curve in Figure 6, shows the mean network connection establishment delay T_{ideal} when all nodes start alternating at the same time t_0. By "ideal" we mean the time where the coordinator is actually elected. The node itself will assume it is the coordinator when its timer expires after time ALT_TIMEOUT. Thus the actual network connection time T_{actual} will be:

$$T_{actual} = T_{ideal} + ALT\_TIMEOUT \quad (6)$$

The curve shows a delay that is increasing slowly with the number of nodes that participate in the network formation. The reason is that there are many one-to-one confrontations occurring in parallel until the coordinator is elected. This is actually a desirable asset of a network establishment protocol. We wouldn't for example like the delay increasing linearly with the number of nodes. We observe that the delay ranges from 1sec to 3sec for a set of nodes that span from N=2 to N=30.

The "no offset" curve yields very small delays partly because all nodes start participating in the network formation at the same time instant. In a more realistic scenario where human users push buttons in order to connect to the network, the nodes will not necessarily start alternating at the same time. We model the "button pushing" as a Poisson process in a W=10sec application window. After the first user, each user i will "arrive" within an iid (truncated) exponentially distributed time L_i, $i = 1,\ldots,N-1$ in the 10sec application window as shown in Figure 7.

The graphs "exp1000" and "exp2000" in Figure 6, show the ideal network formation delay when each user is expected to "arrive" after the first user within 1s and 2s in the average according to the truncated exponential distribution. As the mean value increases, the system becomes more asynchronous and less parallel one-to-one confrontations occur at each time instant. This has an effect of increasing the delay of connection establishment. Nevertheless, the protocol's immunity to the increase of N is preserved. This is illustrated by a constant delay offset between the curves for the same number of N.

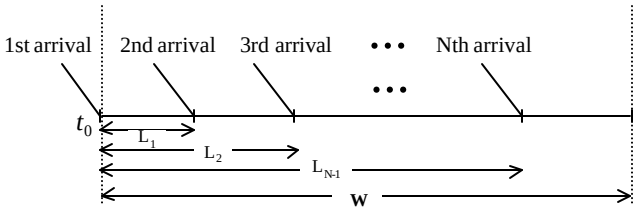

Figure 7: The "push button" arrival process.

The timeout may be viewed as a penalty that has to be paid in order to have a distributed algorithm. A large ALT_TIMEOUT value will satisfy the "correctness" condition with higher probability (higher "timeout efficiency") but will accumulate a larger extra overhead in the actual network connection time T_{actual}. Figure 8 illustrates this trade-off by demonstrating the timeout efficiency as a function of different candidate values of ALT_TIMEOUT[6]. For all application scenarios, the timeout efficiency initially increases rapidly as a function of the timeout and then reaches a steady state. It is clear that the value of ALT_TIMEOUT where the curves start stabilizing, is at 2500ms which is very close to the value 2527.223ms chosen by our empirical formula (5).

The combination of Figures 6 and 8 provide practical guidelines to the designer using the topology construction protocol.

[6] These percentages are the average of the timeout efficiency in the cases of N=5,10,20,30 nodes.

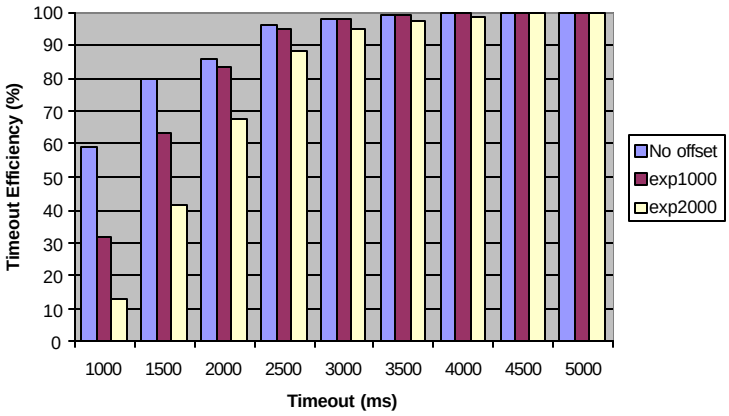

Figure 8: Timeout efficiency for the three conference application scenarios.

For example if there are 30 nodes envisioned participating in the protocol and we choose an ALT_TIMEOUT equal to 2500ms, Figure 6 shows that the average delay experienced by each user will be roughly 3000ms+2500ms=5.500sec and Figure 8 shows that a connected scatternet will be formed with a probability of 96.13% in the case of the "no offset" application scenario.

6. CONCLUSIONS AND DISCUSSION

In ad hoc networks using frequency hopping technology, nodes can be grouped into multiple communication channels. This physical layer setting provides a new way of viewing higher layer functions like topology construction algorithms. Motivated by this environment and using the Bluetooth technology as our research vehicle, we first study the Bluetooth standard asymmetric "sender-receiver" point to point link establishment scheme and then propose a symmetric mechanism for establishing a connection without any role pre-assignment. Based on the ad hoc link formation mechanism we present BTCP, a distributed topology construction protocol where nodes start asynchronously without any prior neighborhood information and result in a network satisfying the connectivity constraints imposed by the Bluetooth technology. The protocol is centered on a leader election process where a coordinator is elected in a distributed fashion and consequently assigns roles to the rest of the nodes in the system.

BTCP was tested under a conference-scenario where users arrive in a room and try to form a scatternet by pressing a "button" on their Bluetooth enabled devices. A nice feature of the protocol is that the network formation delay is sub-linear with the number of participating nodes (implying that the users don't need to wait proportionately longer when more users are present). Although, the delay is small, each node must have an estimate of how long it must participate in the protocol before assuming protocol termination. A conservative estimate of the timeout will introduce unnecessary delays in network formation while an aggressive estimate may leave the network disconnected. Our analysis of the delay statistics of the symmetric link formation protocol provides a tight estimate of the appropriate timeout value, making the protocol fast while ensuring high probability of scatternet connectedness.

Throughout the design of BTCP our aim has been to build a protocol which can be implemented on top of Bluetooth hardware. Although our implemention runs in a Bluetooth emulated environment, when the inter-piconet communication feature is made available in the next release of the Bluetooth hardware, we can test our protocol in an actual setting.

We would like to emphasize that the work presented here is the first approach towards tackling the topology construction problem and providing a fully functional protocol in the Bluetooth frequency hopping environment. There is still much work that remains to be done.

For example, the protocol needs to be extended for the case when not all nodes are within communication range of each other. In this case, after completion of the election process, the coordinator will learn about all participating nodes but not all of them will actually be within its range.

Fortunately, the scatternet can still easily be formed by keeping the election phase I while replacing phases II and III by the following simple procedure: After the coordinator is elected, it pages and connects only to the nodes it confronted and won, since these are the nodes guaranteed to be within its wireless range. Once it has connected to its "one hop" neighbors as a master, it instructs them to start paging, assume the role of masters and repeat the same steps recursively until all nodes are covered. The resulting scatternet is guaranteed to be connected and will have a tree structure rooted at the leader.

Given a set of nodes with zero knowledge of each other that need to form quickly an initial connnected ad hoc network, BTCP focuses on minimizing the connection delay while providing connectedness with high probability. This is a desired property in application scenarios where ad hoc networks continuously connect (birth), perform a coordinated function for a short amount of time (live) and disconnect, since the connection setup delays should be a small fraction of these "birth-live-die" cycles. Keeping this network operation model in mind, alternative methods for topology construction need to be studied and compared in terms of delay with the one presented here.

Finally, in addition to zero-knowledge network initialization, the reformation of an existing network in the face of dynamic changes can be viewed as a separate but equally important issue. After network connection, a separate topology maintenance and optimization protocol needs to run, in order to take care of mobility and/or nodes entering and leaving the network and make sure that the scatternet is reformed accordingly. Such a protocol, although out of the scope of the current paper should be the subject of future resarch efforts.

7. REFERENCES

[1] J. Haartsen, "Bluetooth Baseband Specification, version 1.0", www.Bluetooth.com.

[2] K. Fleming, “Bluetooth Host Controller Interface Functional Specification, version 1.0”, www.Bluetooth.com.

[3] T. Salonidis, P. Bhagwat, L. Tassiulas and R. LaMaire, "Proximity awareness and ad hoc network establishment in Bluetooth", Technical Report, Institute of Systems Research (ISR) University of Maryland, http://www.isr.umd.edu/TechReports.

[4] D. Baker and A. Ephremides, “The architectural organization of a packet radio network via a distributed algorithm”, IEEE Transactions on Communications, COM-29 (1981), pp. 1694-1701.

[5] T.G. Robertazzi, P.E. Sarachik, “Self-Organizing Communication Networks”, IEEE Communications Magazine, Vol 24, No1, Jan 1986.

[6] A. K. Parekh, “Selecting Routers in Ad Hoc Wireless Networks”, SBT/IEEE International Telecommunications Symposium (August 1994).

[7] R.G. Gallager, P.A. Humblet and P.M. Spira, ”A Distributed Algorithm for Minimum Weight Spanning Trees”, ACM Trans. on Program. Lang. and Systems, Vol. 5, pp 66-77, January 1983.

[8] M. Harcol-Balter, T. Leighton, D. Lewin, “Resource Discovery in Distributed Networks”, PODC 1999: p. 229-237.

[9] J. Haartsen, “Bluetooth-The universal radio interface for ad hoc wireless connectivity”, Ericsson Review, no3, 1998.

[10] R. Ramanathan, R. Hain, “Topology Control of Multihop Wireless Networks using transmit power adjustment.”, Proc. Infocom 2000.