Distributed Topology Construction of Bluetooth Wireless Personal Area Networks

Theodoros Salonidis, *Member, IEEE,* Pravin Bhagwat,
Leandros Tassiulas, *Member, IEEE,* and Richard LaMaire

***Abstract*—**

Bluetooth, a wireless technology based on a frequency hopping physical layer, enables portable devices to form short-range wireless ad hoc networks. Bluetooth hosts are not able to communicate unless they have previously discovered each other through synchronization of their timing and frequency hopping patterns. Thus, even if all nodes are within proximity 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, form a connected graph.

Using Bluetooth as an example, we first provide deeper insights into the issue of link establishment in frequency hopping wireless systems. We then introduce an asynchronous distributed protocol that begins with nodes having no knowledge of their surroundings and terminates with the formation of a connected network topology satisfying all constraints posed by Bluetooth. An attractive protocol feature is its ease in implementation using the communication primitives offered by the Bluetooth Specification.

***Index Terms*— Frequency Hopping, Bluetooth, Topology Construction, Scatternet**

I. INTRODUCTION

An ad hoc network is a wireless network formed by nodes that cooperate with each other to forward packets in the network. Most 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 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.

We aim to address a problem that arises when multiple channels are available for communication in an ad hoc network. The problem is determining which subgroup of nodes should share a common channel and which nodes should act as relays, forwarding traffic from one channel to another. The channel assignment should be performed so that all constraints posed by the underlying physical layer are satisfied, while ensuring that the resultant topology is connected.

We address an instance of the above problem which occurs in Bluetooth-based ad hoc networks, known as scatternets. Bluetooth [1] is a promising technology that aims to support 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 time, solutions for point-to-multipoint and multi-hop networking 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 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 protocol to allocate timeslots 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 larger 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 others (M/S bridge) or a slave in multiple piconets (S/S bridge).

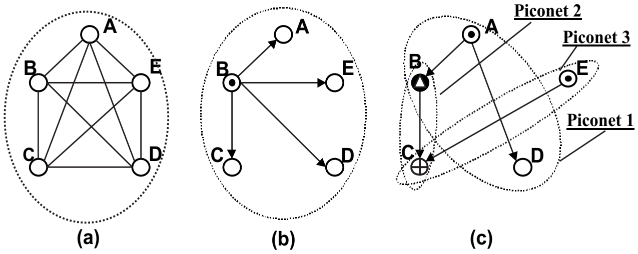

Fig. 1. (a) Single channel topology. (b),(c) Different configurations according to the Bluetooth multi-channel topology model.

It is possible to organize a given set of Bluetooth devices in many different configurations. Figures 1(b) and 1(c) show two

---

This work was supported by an IBM University Partnership Award.

T. Salonidis is with the Department of Electrical and Computer Engineering, Rice University. (e-mail: thsalon@ece.rice.edu).

L. Tassiulas is with the Department of Computer and Communications Engineering, University of Thessaly, Volos Greece and the Department of Electrical and Computer Engineering, University of Maryland, College Park, USA. (e-mail: leandros@inf.uth.gr)

P. Bhagwat is with Wibhu Technologies, the Department of Computer Science and Engineering, I.I.T. Kanpur, India and WINLAB Rutgers University, New Brunswick, USA. (e-mail: pravin@acm.org).

R. LaMaire was with the IBM T.J. Watson Research Center when this work was performed.

example configurations in which nodes in a Bluetooth network can be arranged. All nodes are assumed to be in radio proximity of each other. In Figure 1(b) all nodes are part of a single piconet. Figure 1(c) illustrates another configuration where 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 these scatternet configurations the node interconnection topology in a single channel system will be a complete graph (Fig. 1(a)) since all nodes will hear each other's transmissions.

Given a collection of Bluetooth devices, an explicit topology construction protocol is needed for forming piconets, assigning slaves to piconets, and interconnecting piconets via bridges such that the resulting scatternet is connected. Such a protocol should be asynchronous, distributed and may start with nodes not having any information about their surroundings.

The problem of constructing distributed self-organizing networks has been addressed in the past [2][3][4][5][6][7][8]. All approaches assume existence of a broadcast channel through which neighborhood or control information can become available. The Bluetooth setting introduces two unique challenges: first, no broadcast channel exists for facilitating the exchange of any control information, including proximity information; second, even if proximity information is available, the piconet membership constraint renders the formation of a connected topology a very challenging task.

The scatternet formation problem was introduced in [9] and subsequently addressed in [10] [11] [12] [13] [14] [15] [16] [17][18] [19][20]. Degree-constrained scatternet formation for multi-hop topologies has been investigated in [10][12][13][16]. The problem is NP-complete for some instances and can be solved by a polynomial algorithm under certain assumptions [16]. All proposed solutions in [10][12][13][16] are distributed: starting with the sole knowledge of their one-hop neighbors, the nodes perform role assignments on their adjacent links to reach a connected topology that satisfies the Bluetooth connectivity requirements.

The scatternet formation problem becomes significantly harder if nodes start with no knowledge about their surroundings. The discovery channel is a frequency hopping sequence; nodes in proximity need to synchronize both their timing and frequency hopping patterns before being able to communicate. In this setting, even the formation of individual links becomes an issue—delays are random and can be arbitrarily large if no proper measures are taken.

In this paper (an extended version of [9]) we introduce and analyze a randomized symmetric protocol that yields link establishment delay with predictable statistical properties. Such a protocol is necessary for pairs of identical devices or in situations when any external means for selecting initial device states are not available. We then propose the Bluetooth Topology Construction Protocol (BTCP), an asynchronous distributed protocol that extends the point-to-point symmetric mechanism to the case of several nodes. BTCP is based on a distributed leader election process where proximity information is discovered in a progressive manner and eventually accumulated to an elected coordinator node. Given a view of the topology, the coordinator can then use a centralized algorithm to form a connected scatternet topology.

We present a version of BTCP optimized for the single-hop case (i.e. all nodes are within wireless range of each other). This is a valid assumption for Wireless Personal Area Networks (WPANs), currently considered by the IEEE 802.15 standard [21]. Compared to other forms of ad hoc networks, such as Mobile Ad Hoc Networks (MANETs) or sensor networks, WPANs are characterized by a relatively small number of low-power devices operating within a limited geographic area (e.g. a conference room). In addition to connectivity, WPAN applications require scatternet formation in a short amount of time that is tolerable by a human user.

Zero-knowledge distributed scatternet formation has also been addressed in [11][14][17]. Similar to BTCP, the protocols are distributed and are targeted for single-hop environments. However, they construct and re-arrange the scatternet topology as links are discovered. Bipartite, tree and ring topologies are constructed by the protocols in [11], [14] and [17], respectively. Compared to [11][14][17], BTCP is more flexible in constructing the topology because it uses a centralized algorithm for the role assignment phase.

Apart from scatternet initialization, incremental protocols for scatternet maintenance and reformation have recently been proposed in [18][19]. Liu et. al. consider a specific application context where scatternet formation is triggered by a broadcast route discovery procedure initiated by a source node [20].

The remainder of the paper is organized as follows: Section II introduces the asymmetric link establishment protocol as defined by the Bluetooth Specification. In Section III we propose and analyze the symmetric link establishment protocol. Sections IV and V describe the WPAN application requirements and detailed operation of BTCP, respectively. Since the total number of participants is not known, each node uses a timeout to assume leader election termination. The timeout introduces a correctness-delay tradeoff in the network formation. Using the delay analysis of Section III we show in Section VI how to best choose the protocol parameters in order to maximize the probability of forming a connected scatternet while minimizing delays. Section VII provides a detailed survey of the state-of-the-art in Bluetooth scatternet formation. Section VIII concludes the paper.

II. LINK ESTABLISHMENT IN BLUETOOTH

Bluetooth link establishment is a two-step process that involves the Inquiry and Paging procedures [1]. Both procedures are asymmetric, involving two types of nodes that perform different actions: during Inquiry, senders discover and collect neighborhood information provided by receivers; during Paging, senders connect to previously discovered receivers.

When senders and receivers use the same (Inquiry or Paging) frequency hopping sequence[1], they will most likely start at different frequency hops derived from their local clock readings. To overcome this frequency uncertainty senders and receivers hop at different rates. A receiver changes hops slowly

---

[1] 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.

(every $1.28s$), listening for sender messages; a sender transmits at a much higher rate (every $625\mu s$) while listening in-between transmissions for an answer. The term Frequency Synchronization delay (FS delay) refers to the time needed until the sender transmits on which the frequency the receiver is currently listening[2].

The functional difference between the two procedures is that Inquiry uses a universal frequency hopping sequence while Paging uses a common point-to-point frequency hopping sequence. Using a universal frequency hopping sequence, a sender node effectively broadcasts an Inquiry Access Code (IAC) packet that can be heard by receiver nodes listening for such a packet. During the paging procedure, a sender uses a receiver's page hopping sequence and effectively unicasts a Device Access Code (DAC) packet to be heard only by this receiver. Hence, Inquiry involves many units where a sender can discover more than one receiver while Paging involves only two units where a sender pages and connects to a specific receiver.

A. The Asymmetric Protocol

The asymmetric Bluetooth link establishment protocol (Fig. 2) begins by the sender entering the INQUIRY state and the receiver entering the INQUIRY SCAN state. After an initial FS delay, the sender transmits on the frequency hop the receiver is listening to. Upon reception of the IAC packet, the receiver sleeps for a random time interval (called RB delay), uniformly distributed between 0 and $r_{max}(= 639.375ms)$. The random back-off is performed to avoid collision at the sender in case two or more receivers were listening on the same frequency hop and responded simultaneously.

When the receiver wakes up, it tunes to the hop it was listening before the back-off occurred. After a second FS delay, an IAC packet is received; the receiver replies with an FHS packet and starts listening on its page hopping sequence by entering the PAGE SCAN state. The FHS packet contains the identity and clock of the receiver. Upon reception of the FHS packet, the sender initiates the Paging procedure by entering the PAGE state. The identity and clock in the FHS packet are used to determine the receiver's page hopping sequence and current listening hop, respectively. Thus, when paging follows inquiry, the FS delay is eliminated and the sender transmits a DAC packet on the receiver's listening hop.

The remaining control messages are exchanged in consecutive slots of $625\mu s$ each. The receiver replies with a DAC packet. The sender transmits a FHS packet to let the receiver determine its channel hopping sequence and phase. The receiver acknowledges with another DAC packet and becomes the link slave. As soon as the sender receives the DAC acknowledgment, it becomes the link master. After an additional POLL/NULL packet exchange, the synchronized nodes may start exchanging data.

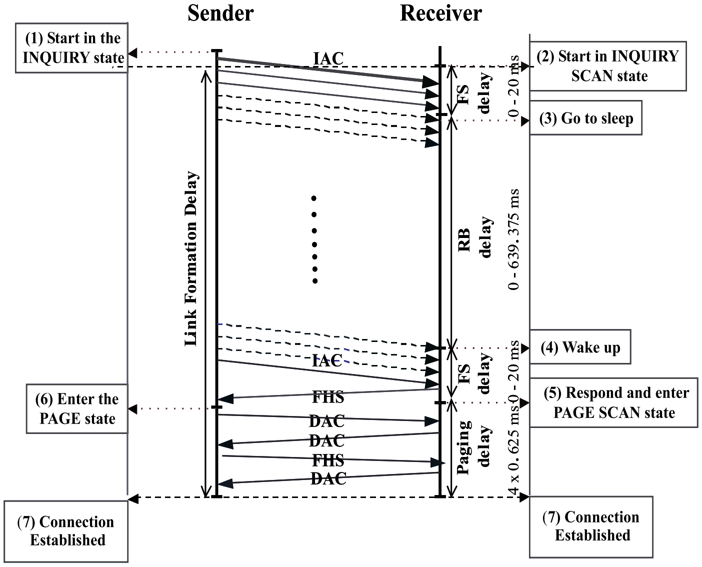

Fig. 2. The Bluetooth asymmetric link establishment protocol.

Figure 2 illustrates the components of the overall protocol delay. The Inquiry delay consists of one RB delay and two FS delays. Since the FS delay is bypassed when paging follows inquiry, paging delay (6 slots, $625\mu s$ each) is assumed negligible. Thus, the overall delay of the asymmetric link establishment protocol can be approximated by:

$$R = 2FS + RB \tag{1}$$

where FS and RB are uniform random variables in $[0, T_{coverage}]$ and $[0, r_{max}]$, respectively.

III. A SYMMETRIC LINK ESTABLISHMENT PROTOCOL

The asymmetric protocol yields a short link establishment delay[3] provided that the sender and receiver roles are pre-assigned. In an ad hoc network setting this may not be possible. For example, in a "conference room" scenario, users are not able to explicitly assign sender and receiver roles on their devices. They just press a button and expect to connect with their peers.

Links can be automatically established using the following symmetric mechanism: When a node is powered on, it arbitrarily assumes sender or receiver role by entering the INQUIRY or INQUIRY SCAN state, respectively. The node remains in the selected state for a period of time. If during this time no connection is established, it switches to the opposite state. State alteration continues until a connection occurs.

Nodes execute the protocol independently; they will be able to connect only during intervals where they are in opposite states. During such an interval, the asymmetric protocol is automatically executed. The sender will become aware of the receiver only when it receives the FHS packet after a random delay R (given by eq. (1)). If during this time the sender independently switches to the receiver state, connection will not occur. On the receiver end, the reception of the IAC packets, back-off activity and transmission of FHS packets are not communicated to the upper layers of the Bluetooth stack. Since we can only have explicit control at the upper layers and since

---

[2] The time needed by the sender to cover the entire inquiry hopping frequency set is $T_{coverage} = N_f \times 625\ \mu s$ which is 10 ms (20 ms) for the 16 (32) hop system.

[3] According to eq. (1), the maximum delay of the asymmetric protocol is $r_{max} + 2 \cdot T_{coverage} = 639.375\ ms + 40\ ms = 679.375\ ms$ for the 32-hop system and 659.375 ms for the 16-hop system.

we need to devise a symmetric protocol without modifying the Bluetooth Specification, we assume that the receiver becomes aware of the sender only after paging and link establishment.

The symmetric protocol operation is depicted in Figure 3. During each ”on” interval X_n, the asymmetric protocol restarts execution. Connection is established only if the generated random delay R_n is less than X_n. Since R is random, the number of ”on” intervals needed until connection will be random. Therefore, the symmetric protocol is expected to have a random delay, typically greater than the delay of the asymmetric protocol.

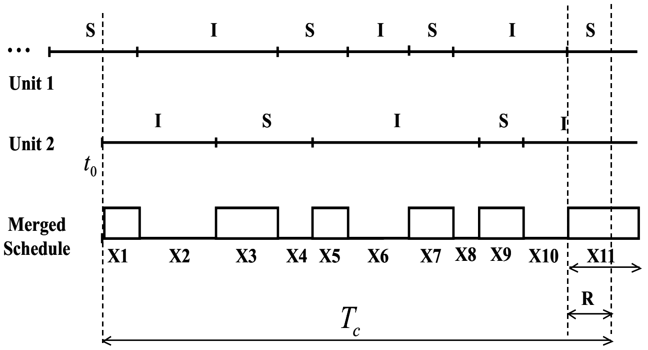

Fig. 3. The symmetric link establishment protocol: Each node alternates independently between INQUIRY (I) and INQUIRY SCAN (S) states. Connection can be established only during the intervals where nodes are in opposite states. The time interval T_c from t_0 up to the point where the two units are in opposite states for a sufficient amount of time is the link establishment delay.

Several interesting questions arise regarding the performance of such a symmetric protocol. Should the state residence intervals be constant or random? How can link establishment delay be minimized?

First, assume the nodes switch states according to a schedule of period T. Since the state residence intervals are constant, the ”on” intervals of the merged process X_n in Figure 3 are also constant. For a specific protocol run, the ”on” intervals can be arbitrarily small and the unsuccessful executions of the asymmetric protocol can be many; the delay, then, will be arbitrarily large. This undesirable phenomenon also holds for an average protocol run (see [22] for a formal proof).

Alternatively, let the state residence intervals form an i.i.d. random process Z_n with mean $E[Z]$ and variance $V[Z]$. If $E[Z]$ and $V[Z]$ are finite, the mean and variance of the link establishment delay T_c are finite and given by:

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

$$V[T_c] = \frac{V[X]}{2} + \frac{(V[X|R>X] + V[X])(1-p)}{p} + V[R] \quad (3)$$

where R is the random link establishment delay of the asymmetric protocol, X_n is the interval process formed by merging the state switching times of the two random alternating schedules, and $p = P[R \leq X]$ (See Appendix I for proof).

Equations (2) and (3) hold for any distribution of finite mean and variance. We have derived the analytical expressions for the cases of exponential and uniform distributions [22]. Figure 4 is a comparative plot of $E[T_c]$ as a function of the mean state residence interval. Both distributions yield U-shaped curves. Very small and very large mean state residence intervals yield high delays. For very small state residence intervals, many short ”on” intervals are needed until connection occurs. For very large state residence intervals, the high delay is due to the uncertainty in the initial state assignment: if the nodes start at the same state, they will wait for a large ”off” interval before the first ”on” interval occurs. The exponential distribution yields a lower delay for large mean state residence intervals. However, both distributions perform similarly in the minimum delay region: for a mean state residence interval of $600\ ms$ the average delay is approximately $1\ s$. This is approximately three times greater than the average delay of the asymmetric protocol given by eq. (1) ($\approx r_{max}/2 = 319.688ms$).

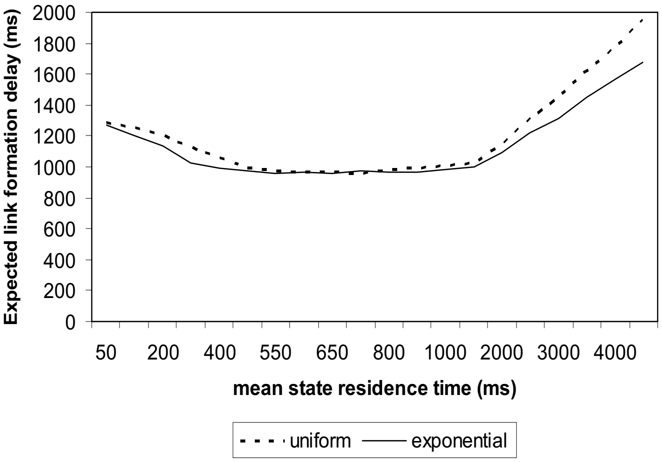

Fig. 4. Symmetric protocol: Average link establishment delay for uniformly and exponentially distributed state residence intervals.

We have also investigated whether using a different mean state residence interval per state yields a lower delay. In this case we use simulations for determining $E[T_c]$. Figure 5 depicts $E[T_c]$ with respect to the INQUIRY mean state residence interval μ_{I}. Each curve corresponds to the INQUIRY SCAN mean state residence interval μ_{S} being $r \times \mu_{\mathrm{I}}$. We observe that there is no benefit in using different mean state residence intervals: In the minimum delay region of all curves the ”$r = 1$” curve yields the lowest average delay.

The randomized symmetric mechanism guarantees automatic link establishment between two Bluetooth devices in finite mean time. When more than two devices need to form a scatternet ”on the fly”, a protocol must be devised on top of this mechanism. This protocol must yield a connected topology with high probability while doing so in minimum time. The delay analysis of the point-to-point symmetric mechanism will provide a valuable tool for balancing these conflicting objectives.

IV. SCATTERNET FORMATION

Our motivation for the scatternet formation problem arises from a ”conference meeting” scenario. Suppose that several

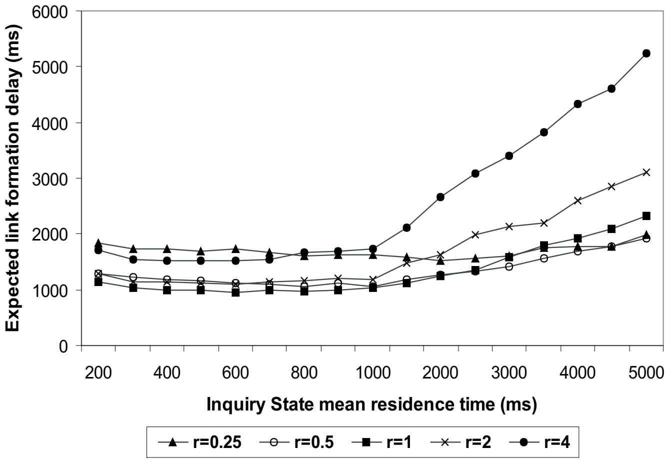

Fig. 5. Symmetric protocol, uniform distribution: Link formation delay for different mean residence intervals per state ($r = \frac{\mu_S}{\mu_I} \neq 1$ curves) vs. link formation delay for equal mean residence intervals per state ($r = 1$ curve).

users wish to form an ad hoc network using their Bluetooth devices. Each user powers on his/her device and expects to see a ”network established” message after a short period of time. After this message appears, the user will be able to exchange information with every other user. The high-level description of this application, embodies the elements of a successful scatternet formation protocol:

- Network establishment must be performed in a distributed manner. Each device must start operating asynchronously on its own without any prior knowledge of the identities or number of nodes participating in the process.
- Network establishment delay must be tolerable by the end-user and minimized as much as possible.
- Upon completion, the protocol must yield a connected scatternet that satisfies the Bluetooth degree constraint of 7 slaves per piconet.

In addition to satisfying connectivity, a desirable protocol feature would be to shape the scatternet topology according to application-specific performance criteria. For example, a node may need to assume different roles in different application scenarios. Also, due to its own nature, a node may pose more restrictive degree constraints: a Palm Pilot may not have the processing power to be a master of a 7-slave piconet. Criteria may also exist in the form of traffic requirements to be satisfied by the nodes participating in the network construction process. The definition of scatternet formation criteria is itself an open research issue that is heavily dependent on the envisioned applications. Our approach takes this issue into account by collecting information about the participating nodes at a single point before the scatternet is actually formed.

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

R1 **Each master will have at most seven slaves:** The constraint posed by the Bluetooth Specification.

R2 **Each node will be either master or slave on *all* its adjacent links:** The Bluetooth Specification does not prevent a node being master in one piconet and slave in others (M/S bridge); However, M/S bridges may result in high delays: when the master visits other piconets as slave, no communication can occur in the piconet it controls. Therefore, we use only S/S bridges to interconnect piconets. Note that with this restriction the resulting topology will be bipartite.

R3 **A bridge node will connect only two piconets:** A bridge node forwards data by switching between piconets in a time division manner. A portable device may have limited processing capabilities. A maximum degree of two relieves the bridge from being an overloaded crossroad of multiple originated data transfers. In addition, the slot overhead incurred by switching multiple piconet time references is minimized [23] [24].

R4 **Every piconet will be connected to *all* other piconets through S/S bridges:** A fully-connected scatternet in its initial state provides higher robustness against topology changes. Also, according to this property, no routing is needed: every master can reach every other master through a bridge node and every slave can reach every other node via its master in at most 3 hops.

R5 **Any two piconets will share only one bridge:** This condition is used for computing the minimum number of piconets and for fast protocol termination. Two masters may later use a topology maintenance protocol to share more than one bridges.

R6 **Given all previous constraints, the scatternet will have the minimum number of piconets:** The motivation for this criterion is similar to finding the minimum number of routers in an ad hoc network [8]: A minimum number of piconets translates to an easier scatternet to control.

V. THE BLUETOOTH TOPOLOGY CONSTRUCTION PROTOCOL (BTCP)

BTCP is based on a leader election process. Leader election is an important tool for breaking symmetry in a distributed system. Since the nodes start asynchronously and without any knowledge of the number of participating nodes, an elected coordinator will be able to control the process and ensure that the resulting topology will satisfy the scatternet formation criteria. The protocol consists of 3 phases:

A. Phase I: Coordinator Election

Phase I consists of an asynchronous distributed election of a coordinator node that will eventually know the count, identities and clocks of all nodes participating in the topology construction process.

Each node has an integer variable called VOTES. Upon power-on, a node initializes VOTES to 1, and starts executing the symmetric link establishment protocol using a randomized schedule.

Any two nodes that discover each other and connect enter a one-on-one confrontation by comparing their VOTES. The node with the larger VOTES wins the confrontation. If the

VOTES are equal, the winner is the node with the larger Bluetooth address. The loser provides the winner with all the FHS packets (i.e. identities and clocks) of the nodes it has won thus far. Then, it disconnects and enters the PAGE SCAN state. In this way, it will hear only page messages from nodes that will page it in the future. This action eliminates the loser from the leader election and prepares it for the next phases of the protocol. Upon receiving the FHS packets, the winner increases its VOTES by the loser VOTES and continues participating in the leader election by resuming execution of the symmetric protocol.

If N nodes are participating in the leader election, there will be $N-1$ confrontations. The winner of the $N-1^{st}$ confrontation becomes the coordinator. At this final state, the rest of the nodes are in the PAGE SCAN state, waiting to be paged by a node that has information about them.

B. *Phase II: Role Determination*

After the election of Phase I, the coordinator has acquired the identities and clocks of all nodes participating in scatternet formation. The coordinator initiates Phase II by checking if the number of nodes N is less than 8. If this is the case, it pages and connects to all other nodes that are waiting in PAGE SCAN; a single piconet is formed with the coordinator as master and the rest of the nodes as slaves. In this special case, the protocol terminates at this point.

If $N \geq 8$, several piconets must be formed and interconnected via bridge nodes. Given the global view of the network, the coordinator can decide on the role each node will perform in the final scatternet. Node-specific scatternet formation criteria can be communicated to the coordinator during the election process as well as the FHS information. Such criteria can be used to derive topologies that satisfy specific optimality objectives in addition to connectivity. Marsan et al. [25] have devised a centralized role assignment algorithm that minimizes the energy consumption of the most overloaded node subject to node traffic requirements.

In absence of specific requirements we will use the default criteria R1-R6. The minimum number of piconets P satisfying R1-R5 is given by:

$$P = \left\lceil \frac{17 - \sqrt{289 - 8N}}{2} \right\rceil, \; 1 \leq N \leq 36 \tag{4}$$

(The proof can be found in [22]). Equation (4) holds for N up to 36 devices; we believe this is a sufficiently large number for the envisioned WPAN application scenarios. Note that this restriction holds if we need to satisfy *all* criteria R1-R6. A larger number of nodes can be supported by either not requiring a minimum P (R6) or by relaxing on one or more of the other criteria (R1-R5).

The coordinator selects itself and $P-1$ nodes as the designated masters and $\frac{P(P-1)}{2}$ other nodes to be S/S bridges. The remaining $N - (P + \frac{P(P-1)}{2})$ nodes are assigned as ”pure” slaves; they are equally distributed among the coordinator and the rest of the masters.

After role assignment, the coordinator constructs for every master X (and itself) a connectivity list set (SLAVESLIST(X), BRIDGELIST(X)). Each list contains contains FHS packets (id+clock) to aid the designated master to page its assigned slaves instantaneously. Next, the coordinator pages and connects to the nodes it selected as masters. (Recall that at the end of Phase I the rest of the nodes wait in the PAGE SCAN state). A temporary piconet is formed with the coordinator as master and the designated masters as slaves[4]. The coordinator transmits to each designated master its connectivity list set and instructs the designated masters to start phase III; then it disconnects the temporary piconet and starts phase III as a master.

C. *Phase III: Connection Establishment*

Phase III is initiated by the designated masters (including the coordinator). Each master pages and connects to the slaves and bridges provided in its SLAVESLIST and BRIDGELIST, 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 (requirement $R3$). When this happens, the bridge node sends a CONNECTED notification to its 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 The protocol operation is depicted in Figure 6.

D. *Leader election termination*

The most time-consuming part of the protocol is the leader election phase. Phases II and III involve only paging and connecting, which happen instantaneously due to the previous inquiry procedures.

Ideally, election should stop as soon as the coordinator is elected. However, since a node is not aware of the total number of participants, it will never know whether or not it is the winner of the election. Each node maintains a ”state alteration” timeout variable called ALT_TIMEOUT. ALT_TIMEOUT is set upon power-on and reset each time the node wins a confrontation and restarts the symmetric link establishment protocol. When ALT_TIMEOUT expires, the node assumes it is the elected coordinator.

It is important to determine an appropriate 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 implies in a very slow Phase I, and a very slow scatternet formation protocol. On the other hand, using a very short ALT_TIMEOUT, several nodes may assume the role of coordinator; this will result in a disconnected scatternet. We address this issue using the following observation: the link formation delay between *any two* out of N alternating nodes is statistically less than the delay of only two alternating nodes. Thus, the delay analysis of the two-node symmetric link establishment protocol can be used to provide a tight estimate for ALT_TIMEOUT.

---

[4] According to eq. (4), P is always less than seven and the temporary piconet can always be formed.

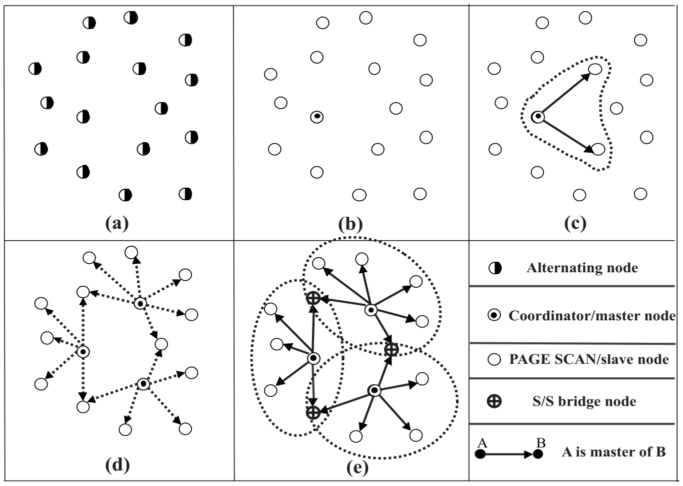

Fig. 6. BTCP operation: (a) Start of Phase I: All nodes start alternating trying to discover other nodes in wireless proximity. (b) End of phase I: Coordinator has been elected. Given N=16 coordinator computes $P = 3$ using eq. (4). Next, the masters, bridges and slaves are selected accordingly. (c) 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) The scatternet is formed.

VI. EXPERIMENTS

A. *Emulating Bluetooth*

We have implemented BTCP on top of an existing prototype implementation that emulates the Bluetooth environment on a Linux platform. The emulator is used instead of actual devices because current Bluetooth hardware does not adequately support scatternet functionality[5]. The emulator also allows testing the protocol for a wide range of parameters and for a large number of nodes.

Each Bluetooth host is implemented as a Linux process consisting 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 [26]. The BTCP module interacts with the BB module through Bluetooth Host Controller Interface (HCI) functions [27]. The use of HCI functions allow us to later replace the BB module with an actual Bluetooth unit.

The wireless medium is simulated by a N_f-hop channel process. The channel process is responsible for the exchange of IAC and FHS packets during the inquiry and paging procedures. It also simulates the occasional frequency collisions and FS delays. Note that the channel process is not similar to a CSMA broadcast channel since the senders and receivers cannot perform any carrier sensing or any form of intelligent back-off. All Bluetooth host processes are connected to the N_f-hop channel process and execute the scatternet formation protocol.

B. *Determining ALT_TIMEOUT*

Using the the Periodic_Inquiry_Mode HCI command [27], it is possible to program Bluetooth units to alternate between INQUIRY and INQUIRY SCAN with uniformly distributed state residence intervals. Figure 7 plots the mean $E[T_c]$ and standard deviation $\sqrt{V[T_c]}$ of the two-node link establishment delay as a function of the mean state residence interval.

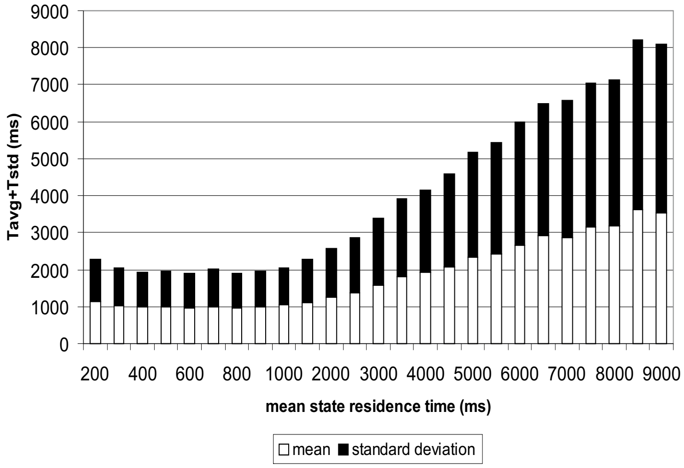

Fig. 7. Nodes alternate with state residence intervals drawn from a uniform distribution of mean μ $msec$. The mean $E[T_c]$ and standard deviation $\sqrt{V[T_c]}$ of the delay of the symmetric protocol for two nodes are plotted as a function of μ.

Given $E[T_c]$ and $V[T_c]$, ALT_TIMEOUT is determined by the following empirical formula:

$$ALT\_TIMEOUT = E[T_c] + \sqrt{V[T_c]} + r_{max} \tag{5}$$

According to Figure 7, for every mean state residence interval, the standard deviation is comparable to the mean. This indicates that the distribution of T_c is not centered around the mean and justifies the inclusion of the term $\sqrt{V(T_c)}$ in eq. (5). The term r_{max} was determined by experimentation. During many protocol runs, the following frequent phenomenon was observed: after the $N-2^{nd}$ confrontation, the winner A would start alternating by resetting ALT_TIMEOUT while another node B was in SLEEP mode due to a previous back-off. A and B were the last nodes in the election process and would start trying to form the $N-1^{st}$ connection only after B woke up. The term r_{max} is the upper bound on the back-off interval of the asymmetric protocol and was included in eq. (5) to take this case into account.

In the experiments we use a mean state residence interval of $600ms$, which, according to Fig. 7 and eq. (5) yields a minimum ALT_TIMEOUT of 2527.223ms.

C. *Protocol Performance*

We use the average scatternet formation delay and the probability of connection as the protocol performance metrics. The scatternet formation delay is dominated by the delay to elect the coordinator (Phase I). Phases II and III are very fast since they involve only paging and connection establishment. Without loss of accuracy we will represent the overall scatternet formation delay by the leader election delay.

We also distinguish between the ”ideal” and ”actual” leader election delays, termed as T_{ideal} and T_{actual}, respectively. T_{ideal} is the delay from the time when the first node is powered-on until the coordinator is elected. It is ideal in the sense that the protocol would terminate at this point had the nodes known

---

[5] Some Bluetooth chipsets from CSR (www.csr.com) provide limited scatternet support–only M/S bridges participating in at most two piconets.

the number of participants; however, a node will assume it is the coordinator after an additional delay of ALT_TIMEOUT. Therefore, the actual scatternet formation delay T_{actual} is given by:

$$T_{\text{actual}} = T_{\text{ideal}} + ALT\_TIMEOUT \tag{6}$$

The probability of connection is the fraction of experiments where only a single node assumes the role of coordinator. This metric depends on the value of ALT_TIMEOUT. The higher ALT_TIMEOUT is, the higher the probability of connection, but the longer the scatternet formation delay.

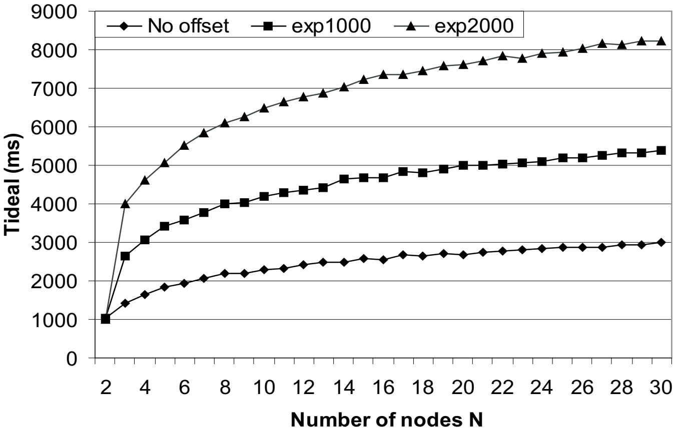

Fig. 8. Average ideal scatternet formation delay for various application scenarios. Units alternate according to uniformly distributed state residence intervals of 600 ms on the average. Each data point is the average of 10,000 runs.

The protocol delay performance is summarized in Figure 8. The "no offset" curve corresponds to T_{ideal} when all nodes start alternating simultaneously. Delay increases with the number of nodes in a sub-linear manner. This is due to the multiple one-on-one confrontations occurring in parallel during the leader election process. This behavior is a desirable property of a scatternet formation protocol. We wouldn't like for example the delay increasing linearly with N. The delay ranges from $1\ s$ to $3\ s$ for $N = 2$ to $N = 30$ nodes.

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 real world scenario, users will power on their devices in an asynchronous manner. We model the power-ons as a Poisson arrival process within a $W = 10\ s$ application window: after the first user, each user i arrives after an exponentially distributed delay L_i of mean μ_p and truncated within the $W = 10\ s$ application window. The truncated exponential distribution is preferred to others (e.g. uniform) because it spreads the arrivals over the entire application window. The process is shown in Figure 9.

The curves "exp1000" and "exp2000" in Figure 8 illustrate T_{ideal} when each user is expected to arrive after the first user within $\mu_p = 1\ s$ and $\mu_p = 2\ s$ on the average, respectively. As μ_p increases, the system becomes more asynchronous and less one-on-one confrontations occur in parallel. This yields an increase in the scatternet formation delay. Nevertheless, the protocol's immunity to the increase of N is preserved. This is illustrated by a constant delay offset between the curves for a fixed N.

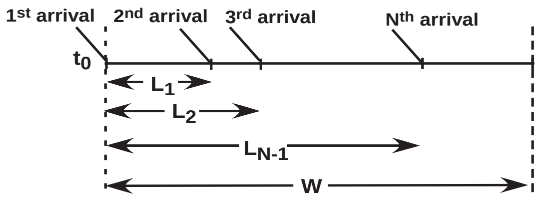

Fig. 9. The device power-on arrival process. The first user arrives at t_0. Each user i arrives after an interval L_i, drawn from a truncated exponential distribution of mean μ_p and upper bound W.

The timeout can be viewed as a delay overhead due to the need for a distributed algorithm. A large ALT_TIMEOUT will yield a connected scatternet with higher probability, but will accumulate a larger actual connection delay T_{actual}. Figure 10 illustrates this trade-off by depicting the probability of connection ("timeout efficiency") for several candidate values of ALT_TIMEOUT. For all application scenarios, the timeout efficiency initially increases rapidly with ALT_TIMEOUT and then reaches a steady state. It is clear that the value of ALT_TIMEOUT where the curves start stabilizing is at $2500\ ms$–very close to the value $2527.223\ ms$ chosen by our empirical formula (eq. (5)).

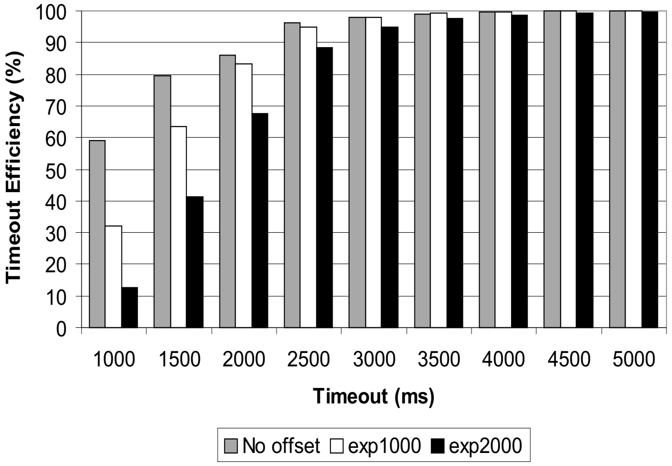

Fig. 10. Timeout efficiency: Each bar graph is the probability of connection, averaged over N=5,10,20 and 30 nodes (10000 runs for each N).

When an upper bound estimate exists on the number of nodes participating in the protocol, the combination of Figures 8 and 10 provides practical guidelines. For example, if the expected number of nodes is 30 and an ALT_TIMEOUT of $2500ms$ is used, the average delay experienced by each user will be $3000ms + 2500ms = 5.5s$ (Fig. 8) and a connected scatternet will be formed with a probability of 96.13% in the "no offset" application scenario (Fig. 10).

VII. RELATED WORK

The scatternet formation problem can be summarized as follows: "Given the network visibility graph induced by the nodes' wireless proximity, establish a subset of master/slave links such that the resulting communication graph is connected and satisfies the Bluetooth degree constraints".

Using a Minimum Spanning Tree (MST) framework, Guerin et. al. [16] show that the scatternet formation problem is NP-complete for general visibility graphs[6]. When nodes are distributed on a 2-dimensional plane (Euclidean visibility graphs), the problem can be solved by a MST construction algorithm of polynomial complexity[7]. This is because every node belonging to a Euclidean MST has at most 6 adjacent links–less than the Bluetooth constraint of 7.

Most proposed solutions to the scatternet formation problem are distributed. The protocols can be classified according to the initial information available to the nodes and the structure of the generated topologies.

In [10][16][12] [13][15] the nodes start with a-priori knowledge of their one-hop neighbors. Záruba et. al. [10] present a protocol for Euclidean visibility graphs where a designated root node initiates scatternet formation and forms a tree topology. A geometric argument[8] is used to re-assign roles on links in case some nodes exceed the degree constraints during the formation process. Guerin et. al. [16] propose a heuristic of low communication complexity for construction of tree topologies in Euclidean visibility graphs. The approach requires knowledge of the neighbor coordinates, provided by GPS hardware at each node.

Li and Stojmenovic [15] generate connected non-tree scatternet topologies for the Euclidean case. The protocol applies Yao structure, which requires knowledge of the neighbor coordinates. Petrioli and Basagni [13] trade off the cost of extra GPS hardware by extending the required initial knowledge to two hops. They combine clustering techniques with the geometric argument of [10] to yield connected non-tree scatternet topologies. Bluenet [12] is a heuristic protocol for the construction of non-tree scatternet topologies from arbitrary (non-Euclidean) visibility graphs. A comparative performance evaluation of some of the above protocols can be found in [28].

The problem certainly does not become easier when the nodes start with no knowledge about their surroundings. Due to the random discovery delays it is hard to make any deterministic claims regarding connectivity, even for the Euclidean case. It is not straightforward to extend the multi-hop protocols in [10][16][12] [13][15] to the zero-knowledge setting because they assume static topologies and do not operate in an incremental manner.

On the other hand, BTCP and the protocols in [11][14][17], are targeted for the zero-knowledge setting but are currently restricted to the single-hop environment (the visibility graph is complete). Law et. al. [11] and Zhang et. al. [17] aim at constructing bipartite and ring topologies, respectively. It is analytically shown that the resulting topologies will be connected with high probability. Both protocols operate in synchronous rounds of fixed length where nodes assume sender and receiver roles with a certain probability. The round length is assumed sufficiently large to guarantee connection of two nodes that start in opposite states. However, synchronous operation is difficult to support in a zero-knowledge setting. Tan et. al. [14] propose an asynchronous incremental protocol that creates tree fragments, continuously merged to yield a single tree topology. BTCP is both distributed and asynchronous; it also provides more flexibility in forming the final WPAN topology due to its centralized role assignment phase.

Incremental scatternet formation for multi-hop visibility graphs has recently been considered in [18][19]. In [18] each node uses a set of local role assignment rules on discovered links to maintain the Bluetooth degree constraint. The local rules do not target global scatternet connectivity. In [19], Cuomo et. al. focus on the creation of tree topologies by extending [14] for multi-hop euclidean visibility graphs. Degree constraints are taken into account using the geometric argument of [10]. Connectivity properties of the resulting tree topologies are evaluated through simulations.

A recent work by Liu et. al. [20] considers multi-hop scatternet formation within a specific application context: topology discovery and construction are triggered by a broadcast route discovery procedure initiated by a source node. This application requires the source be aware of the destination. Modifications to the Bluetooth specification are proposed for senders being able to broadcast their address during inquiry. Initial inquiry states are assumed preassigned: the source node starts as sender while the rest start as receivers.

VIII. CONCLUSIONS AND DISCUSSION

In ad hoc networks using frequency hopping technology, nodes can be grouped in multiple communication channels. This physical layer setting provides a new way of viewing higher layer functions like topology construction. 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 turning on their Bluetooth-enabled devices. An attractive 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

---

[6] NP-completeness holds if a node is forced to act as master or slave to all its adjacent links. It has not been determined whether the same result holds if M/S bridges are allowed.

[7] The MST is constructed by considering as edge weights the distances between nodes in the visibility graph

[8] In a Euclidean graph, if a node has more than 5 neighbors, then at least two of them are within wireless proximity of each other.

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 was to build a protocol which can run on top of Bluetooth hardware. Our modular implementation runs in a Bluetooth emulated environment; it can later be used on Bluetooth devices with adequate support of scatternet functionality.

The protocol needs to be extended for the multi-hop case. The leader election mechanism can serve as a building block for discovering, connecting partial topology views and then merging them in larger components. A possible implementation of this idea is as follows: During the election process a node maintains a topology map in addition to the FHS packets of the nodes it has won so far. After a one-on-one confrontation, the loser communicates its FHS packets and topology map to the winner. Before starting alternating, the winner pages the nodes indicated in the loser topology map. (Temporary) connections will be established only with the paged nodes that are within proximity of the winner. This results in the winner node updating its local topology map; this process continues until the node loses a one-on-one confrontation or becomes the coordinator. The coordinator uses a centralized algorithm to produce an optimized scatternet based on the discovered topology graph. Using this modified leader election mechanism, it is likely that multiple leaders will be elected and form scatternet clusters with no nodes in common. The clusters are further discovered and merged using a new leader election process operating at the cluster level.

Given a set of nodes with zero knowledge of each other that need to form quickly an initial connected 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(die); 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.

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 need to be run to accommodate mobility and/or nodes entering and leaving the network while ensuring that the scatternet is reformed accordingly. Such a protocol should be the subject of future research efforts.

APPENDIX I
PROOF OF EQUATIONS (2) AND (3)

Let $N_i(t)$ be the number of state switches of node i from time t_0 up to time t (Fig. 3). $N_i(t)$ is a renewal process induced by the i.i.d. interval process Z_n. Since the units alternate independently, $N_1(t)$ and $N_2(t)$ are independent. Let $N(t)$ be process resulting from the merged state switches of $N_1(t)$ and $N_2(t)$. The interval process X_n induced by $N(t)$ is i.i.d. [29].

Case A: Let t_0 be such that the nodes start in opposite states. In this case, for each n odd, connection will occur only if the delay R_n of the asymmetric protocol is less than X_n. Since each of R_n and X_n are i.i.d. and independent with respect to each other, this is equivalent to a coin-toss experiment with probability of ”connection-success” $p = P[X \leq R]$. Let the composite (”on”+”off”) interval Y_n corresponding to a ”failure” be defined as:

$$Y_n = \begin{cases} X_n + X_{n+1} & \text{if } R_n > X_n \\ 0 & \text{otherwise} \end{cases}, \; n = 2k+1, \; \forall k \geq 0 \quad (7)$$

The overall connection establishment delay T_c^{opp} when nodes start in opposite states is:

$$T_c^{opp} = \sum_{n=1}^{N} Y_n + R_{N+1} \quad (8)$$

where N is the number of failures until a success occurs and is geometrically distributed with parameter $p = P[R \leq X]$. Thus, the average delay can be computed as follows:

$$\begin{aligned} E[T_c^{opp}] &= E[E[T_c^{opp}]|N] + E[R] \\ &= \sum_{n=0}^{\infty} E[T_c^{opp}|N = n] \cdot P[N = n] + E[R] \\ &= \sum_{n=0}^{\infty} E[\sum_{i=1}^{n} Y_i] \cdot P[N = n] + E[R] \\ &= \sum_{n=0}^{\infty} n \cdot E[Y_i] \cdot P[N = n] + E[R] \\ &= E[Y_i] \cdot E[N] + E[R] \\ &= (E[X|R > X] + E[X]) \cdot E[N] + E[R] \Rightarrow \end{aligned}$$

$$E[T_c^{opp}] = \frac{(E[X|R > X] + E[X])(1-p)}{p} + E[R] \quad (9)$$

Case B: Let t_0 be such that the nodes start at the same state. The only difference with case A is that the first ”off” interval introduces a constant delay factor on the overall delay. Therefore:

$$T_c^{same} = X + T_c^{opp} \quad (10)$$

where T_c^{opp} is given by eq. (8). Then,

$$E[T_c^{same}] = E[X] + E[T_c^{opp}] \quad (11)$$

Since t_0 is arbitrary, cases A and B are equiprobable. Thus, $E[T_c] = \frac{1}{2}E[T_c^{opp}] + \frac{1}{2}E[T_c^{same}]$. Using eq. (9) and eq. (11), we reach eq. (2) for $E[T_c]$.

To derive the variance $V[T_c]$, observe that eq. (8) and eq. (10) are sums of independent random variables. Therefore, the linearity of variance holds in the same way as linearity of expectation. Repeating the same calculation as in $E[T_c]$, we reach the desired eq. (3) for $V[T_c]$.

REFERENCES

[1] BluetoothSIG, “Specification of the Bluetooth system, version 1.2,” in *www.bluetooth.com*.

[2] D. Baker and A. Ephremides, “The architectural organization of a packet radio network via a distributed algorithm,” *Proc. IEEE Transactions on Communications*, vol. 29, pp. 1694–1701, 1981.

[3] A. Ephremides, J. Wieselthier, and D. Baker, “A design concept for reliable mobile radio networks with frequency hopping signaling,” *Proc. of IEEE*, vol. 75, no. 1, pp. 56–73, 1987.

[4] T. Robertazzi and P. Sarachik, “Self-Organizing Communication Networks,” *IEEE Communications Magazine*, vol. 24, no. 1, 1986.

[5] M. Gerla and T. Tsai, “Multicluster, mobile multimedia radio network,” *Proc. ACM Baltzer Journal of Wireless Networks*, vol. 1, pp. 255–65, August 1995.

[6] C. Lin and M. Gerla, “Adaptive Clustering for Mobile Wireless Networks,” *Proc. IEEE Journal on Selected Areas in Communications*, vol. 15, no. 7, pp. 1265–1275, 1997.

[7] R. Ramanathan and R. Hain, “Topology Control of Multihop Radio Networks using Transmit Power Adjustment,” in *Proc. IEEE INFOCOM*, Tel Aviv, Israel, March 2000.

[8] A. Parekh, “Selecting Routers in Ad Hoc wireless networks,” in *SBT/IEEE International Telecommunications Symposium*, Acapulco, Mexico, August 1994.

[9] T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire, “Distributed Topology Construction of Bluetooth Personal Area Networks,” in *Proc. IEEE INFOCOM*, Anchorage, AK, April 2001.

[10] G. Záruba, S. Basagni, and I. Chlamtac, “Bluetrees - scatternet formation to enable Bluetooth-based ad hoc networks,” in *Proc. International Conference on Computer Communications (ICC)*, St. Petersburg, Russia, June 2001.

[11] C. Law, A. Mehta, and K. Siu, “Performance of a new Bluetooth scatternet formation protocol,” in *Proc. ACM MOBIHOC*, Long Beach, CA, October 2001.

[12] Z. Wang, R. Thomas, and Z. Haas, “Bluenet-a New Scatternet Formation Scheme,” in *Proc. HICSS*, Big Island, Hawaii, January 2002.

[13] C. Petrioli and S. Basagni, “Degree-Constrained Multihop Scatternet Formation for Bluetooth Networks,” in *Proc. GLOBECOM*, Taipei, Taiwan, November 2002.

[14] G. Tan, A. Miu, J. Guttag, and H. Balakrishnan, “An Efficient Scatternet Formation Algorithm for Dynamic Environments,” in *Proc. IASTED Communications and Computer Networks (CCN)*, Cambridge, MA, November 2002.

[15] X. Li and I. Stojmenovic, “Partial Delaunay triangulation and degree-limited localized Bluetooth scatternet formation,” in *Proc. AD-HOC NetwOrks and Wireless (ADHOC-NOW)*, Toronto, Canada, September 2002, pp. 17–32.

[16] R. Guerin, J. Rank, S. Sarkar, and E. Vergetis, “Forming Connected Topologies in Bluetooth Adhoc Networks,” in *Proc. International Teletraffic Congress (ITC)*, Berlin, Germany, September 2003.

[17] H. Zhang, J. Hou, and L. Sha, “A Bluetooth Loop Scatternet Formation Algorithm,” in *Proc. International Conference on Computer Communications (ICC)*, Anchorage, AK, May 2003, pp. 1174–1180.

[18] C. Chiasserini, M. Ajmone-Marsan, E. Baralis, and P. Garza, “Towards Feasible Distributed Topology Formation Algorithms for Bluetooth-based WPANs,” in *36th Hawaii International Conference on System Science (HICSS-36)*, Big Island, HI, January 2003.

[19] F. Cuomo, G. DiBacco, and T. Melodia, “SHAPER: a self-healing algorithm producing multi-hop bluetooth scatternets.” in *Proc. GLOBECOM*, San Fransisco, CA, December 2003, pp. 236–240.

[20] Y. Liu, M. Lee, and T. Saadawi, “A Bluetooth Scatternet-route Structure for Multihop Ad Hoc Networks,” *Proc. IEEE Journal on Selected Areas in Communications*, vol. 21, pp. 229–239, February 2003.

[21] I. . W. G. for WPAN, “IEEE 802.15 Std. Specification, 2002,” in *http://grouper.ieee.org/groups/802/15/*.

[22] 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*, vol. TR 2001-10, p. http://www.isr.umd.edu/TechReports, 2001.

[23] G. Miklos, Z. Turanyi, A. Valko, and P. Johansson, “Performance Aspects of Bluetooth Scatternet Formation,” in *Proc. ACM MOBIHOC*, Boston, MA, August 2000.

[24] T. Salonidis and L. Tassiulas, “Performance issues of Bluetooth scatternets and other asynchronous TDMA ad hoc networks,” in *Proc. International Workshop on Mobile Multimedia Communications (MoMuC)*, Munich, Germany, October 2003.

[25] M. Marsan, C. Chiasserini, A. Nucci, and G. Carello, “Optimizing the topology of bluetooth wireless personal area networks,” in *Proc. IEEE INFOCOM*, New York City, NY, June 2002.

[26] BluetoothSIG, “Bluetooth Baseband Specification, version 1.2,” in *www.bluetooth.com*.

[27] ——, “Bluetooth Host Controller Interface Functional Specification, version 1.2,” in *www.bluetooth.com*.

[28] S. Basagni, R. Bruno, and C. Petrioli, “A Performance comparison of Scatternet Formation Protocols for Networks of Bluetooth Devices.” in *Proc. IEEE Conference on Pervasive Computing and Communications (PerCom)*, Dallas Fort-Worth, TX, March 2003, pp. 341–350.

[29] K. Trivedi, *Probability and Statistics with reliability queuing and Computer Science Applications*.

Theodoros Salonidis (S’98-M ’04) is a Post-doctoral Research Associate in the Department of Electrical and Computer Engineering at Rice University. He received the Diploma in Electronic and Computer Engineering from the Technical University of Crete, Greece in 1997 and the M.S. and Ph.D. degrees in Electrical and Computer Engineering from the University of Maryland, College Park in 1999 and 2004, respectively. During one year (1999-2000) he was a Research Intern at IBM T.J. Watson Research Center, New York. His current research interests include resource allocation, topology control, and Quality of Service provisioning in wireless networks. He is a member of IEEE, ACM, and the Technical Chamber of Greece.

Pravin Bhagwat is an entrepreneur and a researcher in the area of wireless and mobile networking. He is the Co-founder and Chief Technology Officer of Wibhu Technologies, a wireless networking startup based in Pune and Sunnyvale, California. Pravin also holds adjunct faculty appointments at computer science department, IIT Kanpur and at WINLAB, Rutgers University, New Jersey. Prior to founding Wibhu Technologies, Pravin worked as the principal architect at a wireless networking startup based in New Jersey and as a technology consultant in the Networking Research group at AT&T Research. Pravin started his research career at IBM Thomas J. Watson Research Center, New York. He actively serves on program committees of networking conferences and has published numerous technical papers and patents in the area of mobile computing and wireless communication. He is the Associate Editor of IEEE Transactions on Mobile Computing and has also served as a guest editor of IEEE Network. Pravin has a B.Tech. in Computer Science from IIT Kanpur and an MS/PhD in computer science from the University of Maryland, College Park, USA.

Leandros Tassiulas (S’89, M’91) was born in 1965, in Katerini, Greece. He obtained the Diploma in Electrical Engineering from the Aristotelian University of Thessaloniki, Thessaloniki, Greece in 1987, and the M.S. and Ph.D. degrees in Electrical Engineering from the University of Maryland, College Park in 1989 and 1991 respectively. He is Professor in the Department of Computer and Telecommunications Engineering, University of Thessaly, Greece and Research Professor in the Department of Electrical and Computer Engineering and the Institute for Systems Research, University of Maryland College Park since 2001. He has held positions as Assistant Professor at Polytechnic University New York (1991-95), Assistant and Associate Professor University of Maryland College Park (1995-2001) and Professor University of Ioannina Greece (1999-2001). His research interests are in the field of computer and communication networks with emphasis on fundamental mathematical models, architectures and protocols of wireless systems, sensor networks, high-speed internet and satellite communications. Dr. Tassiulas received a National Science Foundation (NSF) Research Initiation Award in 1992, an NSF CAREER Award in 1995 an Office of Naval Research, Young Investigator Award in 1997 and a Bodosaki Foundation award in 1999 and the INFOCOM ‘94 best paper award.