DISTRIBUTED CONSTRUCTION AND MAINTENANCE OF BANDWIDTH-EFFICIENT BLUETOOTH SCATTERNETS

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Metin Tekkalmaz

August, 2004

ABSTRACT

DISTRIBUTED CONSTRUCTION AND MAINTENANCE OF BANDWIDTH-EFFICIENT BLUETOOTH SCATTERNETS

Metin Tekkalmaz
M.S. in Computer Engineering
Supervisor: Assist. Prof. Dr. İbrahim Körpeoğlu
August, 2004

Bluetooth is currently the mainstream technology used for short range wireless communication due to its low power and low cost properties. In order to communicate, Bluetooth enabled devices can form networks called piconets, which consist of at most eight members. To construct larger Bluetooth networks, which are called scatternets, any number of piconets can be combined. Although piconet construction process is standardized by Bluetooth Special Interest Group, scatternet construction policies and algorithms are not yet clarified.

There have been many solution proposals for the scatternet construction problem each of which focuses on different aspects of it like the efficiency of the construction algorithm, ease of routing in the resulting scatternet and number of piconets that constitute it. Although various considerations came into picture, bandwidth efficiency of the resulting scatternet topology, which depends on the placement of nodes and communication demand among them, did not take much attention.

In this thesis, we provide a distributed and adaptive algorithm that constructs a scatternet and based on collected traffic flow information, modifies it to minimize the overall bandwidth usage. As consequences of efficient use of available bandwidth, reduce in average latency and total energy consumption as well as increase in available bandwidth for new communication demand are also aimed. Moreover, performance of the proposed algorithm is presented, based on the evaluation criteria described.

Keywords: Bluetooth, Scatternet Construction, Bandwidth-Efficient Topologies.

ÖZET

BANT GENİŞLİĞİNİ VERİMLİ KULLANAN BLUETOOTH SERPME AĞLARININ DAĞITIK OLUŞTURULMASI VE BAKIMI

Metin Tekkalmaz
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Yrd. Doç. Dr. İbrahim Körpeoğlu
Ağustos, 2004

Bluetooth, düşük enerji tüketimi ve düşük maliyeti dolayısı ile, şu anda, kısa mesafeli kablosuz iletişim için kullanılan başlıca teknolojidir. Haberleşme sağlayabilmek için Bluetooth donanımlı cihazlar, en fazla sekiz düğümden oluşan ve *piconet* adı verilen ağlar kurabilmektedirler. *Scatternet* adı verilen daha büyük Bluetooth ağlarını oluşturmak için *piconet*ler birleşebilirler. *Piconet* oluşturma yöntemi için Bluetooth Special Interest Group tarafından bir standart belirlenmiş olmasına rağmen *scatternet* oluşturma yöntemleri henüz tanımlanmamıştır.

Şu ana kadar, oluşturma işlemininin verimliliği, oluşturulan *scatternet*te yol atama kolaylığı ya da *scatternet*i meydana getiren *piconet* sayısının asgariye indirmesi gibi farklı ana hedefleri olan çok sayıda *scatternet* oluşturma yöntemi önerilmiştir. Önerilen yöntemlerde farklı bir çok nokta göz önünde bulundurulmuş olmasına rağmen bant genişliğinin verimli kullanılması konusu yeterli dikkati çekememiştir.

Bu tezde, toplam bant genişliği kullanımını asgariye indirmeyi amaçlayan dağıtık ve değişen koşullara uyum sağlayabilen bir *scatternet* oluşturma ve idame yöntemi önerilmektedir. Bant genişliğinin verimli kullanımına bağlı olarak, ortalama paket gecikmesi ve toplam enerji tüketiminde düşüşün yanı sıra yeni haberleşme ihtiyaçları için kullanılmayan bant genişliğini arttırmak da hedeflenmektedir. Değerlendirme kriterleri ve bunlara dayalı olarak, önerilen yöntemin başarımı da tez çalışmasında sunulmaktadır.

Anahtar sözcükler: Bluetooth, Scatternet (Serpme Ağ) Oluşturma, Bant Genişliğini Verimli Kullanan İlingeler.

To my grandfather being the first among very few people I admire...

Acknowledgement

I would like to express my gratitude to my supervisor Assist. Prof. Dr. İbrahim Körpeoğlu for his realistic, encouraging and constructive approach throughout my masters study and his efforts during supervision of the thesis.

I would like to thank to Prof. Dr. Özgür Ulusoy and Assist. Prof. Dr. Uğur Güdükbay for kindly accepting to spend their valuable time and for evaluation of my thesis.

I would like to express my special thanks to Hasan Sözer for his cooperation to make this thesis work come true and for his friendship in daily life.

I thank to the members of Networking and Systems Group at Bilkent University during the years 2002, 2003 and 2004 for their comments on my studies and for the sparks they caused. I also thank to the Scientific and Technical Research Council of Turkey (TÜBİTAK) for its support to the project with grant number 103E014.

I would like to express my appreciation to ASELSAN A.Ş. for the understanding and support during my academic studies. I also want to thank to people I work with for the joy they bring to my life making my business life together with academic life a pleasureful experience.

Finally, I would like to express my thanks to my parents, making me who I am now with their love, trust, freedom understanding and every kind of support throughout my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Bluetooth [4] is a short range wireless RF technology designed initially for cable replacement at indoor places, but also supports usage scenarios for personal area or local area networking. Its low cost, low power consumption and ad-hoc connectivity features make it a good wireless connectivity choice for mobile devices due to their well-known characteristics such as limited battery power and anytime-anywhere connection requirements.

Bluetooth enabled devices are able to form small networks, which are called piconets, with up to eight nodes per piconet. A piconet consists of a master node and one or more slave nodes. The scheduling of packets into the common Frequency Hopping Spread Spectrum (FHSS) radio channel in a piconet and accessing this common radio channel is coordinated by the master node via a polling based Time Division Multiple Access (TDMA) scheme. No direct communication between any two slave nodes is allowed in a single piconet and data traffic among slave nodes has to go through the master node.

Bluetooth devices have capability of forming piconets but are not restricted with them. Bluetooth standards specify some set of mechanisms to construct larger Bluetooth networks called scatternets. In this way, the 8-node limit in a Bluetooth network can be overcome. A scatternet is actually nothing but a network consisting of multiple Bluetooth piconets with common nodes to forward traffic between piconets they belong to. Such an intermediate node is called as bridge. Since all members of a single piconet follows the same pseudorandom hopping sequence over some set of RF channels specified in Bluetooth standards, and since each piconet has a different hopping sequence, a bridge node is frequency-synchronized with (i.e. follows the hopping sequence of) different piconets, which it belongs to, at different times.

Although piconet formation is a well-defined process described in current Bluetooth standards, scatternet construction algorithms and policies are not specified in detail, even though there are mechanisms specified that will enable construction of scatternets. Therefore, this is an open to research area and many methods have been proposed so far for scatternet construction problem. Although the goal is the same, which is to construct a scatternet, different proposals focus on different aspects, such as constructing the scatternet in a small amount of time, minimizing the number of messages required during construction, making the resulting scatternet easy to route on, or minimizing the number of piconets that constitute the resulting scatternet, etc. Once the scatternet is constructed, maintaining it, when existing nodes leave and new nodes join, is another issue, which is not addressed much in the literature. Also, to the best of our knowledge, there is a very limited number of studies [2, 15, 24] that focus on constructing a scatternet with the aim of utilizing the bandwidth capacity of the resulting scatternet as efficiently as possible, considering the traffic demands of nodes making the scatternet.

In this thesis, we propose a scatternet construction algorithm that also handles arriving and leaving nodes and when necessary modifies the topology to make it bandwidth-efficient. Bandwidth-efficiency is preserved as the node set constituting the scatternet and the communication demands among them change. Given a scatternet, which has a bandwidth capacity depending on its topology, there are two main factors which affect the level of usage of its capacity:

- End-to-end traffic demands (i.e. flow rates) among nodes
- Number of hops that the data packets traverse as they go from source to destination (i.e. path lengths)

Since the traffic demand should be satisfied, number of hops between nodes is the only parameter to be altered. In order to achieve our goal, which is to reduce the load that communication demands impose on the network, we reduce the number of hops between nodes that have relatively more communication demand as our basic approach, similar to [24]. But different from [24] our algorithm works in a distributed manner with no central coordination and without requiring the availability of global topology and traffic demand information *a priori*. Although it is a distributed algorithm, the messaging overhead to construct and maintain the scatternet is kept as low as possible. Our algorithm is also adaptive to node departures, node arrivals and changes in traffic demands between nodes, so that the scatternet remains bandwidth efficient despite the dynamic nature of the scatternet topologies consisting of mobile nodes and dynamic nature of traffic flows.

The basic approach, which is used to obtain efficient topologies with respect to bandwidth usage, and its benefits can better be understood with some figures. Assume that there is a scatternet containing nodes A, B, C, D, E and F, where C and D are the master nodes of the two piconets, which constitute the scatternet, and E is the bridge node in between initially, as shown in Figure 1.1 (a). Let A-B and C-D pairs communicate at the rates of d and $2d$ bits per second respectively. The load on the scatternet due to these communication demands is shown in Figure 1.1 (b). A-B communication goes through four links, whereas C-D communication traverses two links, which make the total load generated by these two communications $4 \times d + 2 \times 2d = 8d$. As it can be observed from Figure 1.1 (b), nodes C, D and E are involved in A-B communication, although the flow is not directly related with them. Similarly, E is used as an intermediate node in C-D communication. Since the communicating nodes are distant from each other, as far as the hop distances are concerned, following problems are faced: 1. The bandwidth of the scatternet is wasted, 2. Energy of some nodes is drained for the traffic that is not related with them. Now consider a slightly changed version of the topology in Figure 1.1 (a) and (b) where locations and roles of nodes B and C are *swapped* as shown in Figure 1.1 (c). Problems listed previously are eliminated, since, the load imposed by communication demands

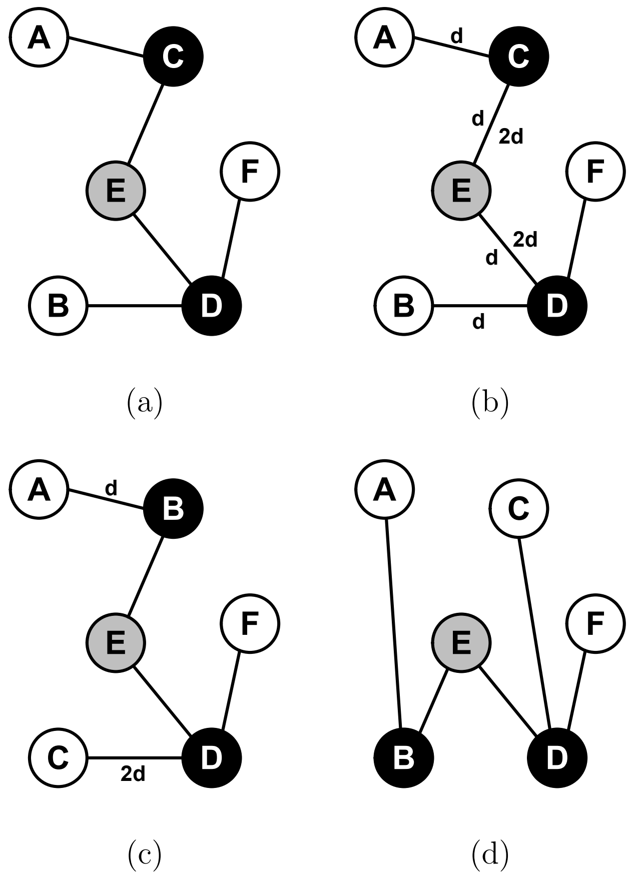


Figure 1.1: Basic approach. (a) Initial topology. Loads on links for (b) initial topology and (c) altered topology. (d) Breaking and establishing links to get desired topology.

described above is now $3d$ and no intermediate node over communication paths is used. Furthermore, latency is decreased, since the number of intermediate nodes used for forwarding the traffic is reduced. Swapping locations mentioned above is actually not a physical location change but a process in which the new topology is obtained by breaking some links and establishing new ones. For example, topology in Figure 1.1 (d) - and also topology in Figure 1.1 (c), since graphs representing topologies in these figures are isomorphic - can be obtained from topology in Figure 1.1 (a) by first breaking A-C, C-E and B-D links, then establishing A-B, B-E and C-D links and finally assigning appropriate master/slave roles to the required nodes.

Along with the details of the proposed algorithm itself, its evaluation is also presented in the thesis. In order to evaluate the algorithm a custom simulation environment is prepared and effects of the algorithm on the bandwidth-efficiency of the scatternets are examined. The simulation is run for various scatternet sizes and different traffic demand characteristics. According to the observations of simulation results, the proposed algorithm is useful especially for scatternets with uneven, diverse traffic demand characteristics and for scatternets with groups which consist of nodes having higher tendency to communicate within the group. Simulation results also shows that the algorithm provides improvements up to about 45% in bandwidth-efficiency according to the Weighted Average Shortest Path criterion.

The remainder of the thesis is organized as follows. In the next chapter, necessary background information about Bluetooth is provided, which is followed, in Chapter 3, by a discussion of related studies in the literature. In Chapter 4, a detailed description of the proposed scatternet construction algorithm is given and in Chapter 5, evaluation criteria for bandwidth efficiency is explained, simulation environment is described and the results are presented. Finally, the thesis is concluded in Chapter 6, along with some future work issues.

Chapter 2

Background Information

This chapter gives information about Bluetooth technology to provide background before explaining the proposed scatternet construction algorithm in detail. The subsequent sections are intended to give both the necessary technical information to make the further discussions clear and the motivation behind the scatternet construction in general. Information presented in this chapter is mainly compiled from [4, 3, 11, 5, 1].

2.1 Bluetooth

Bluetooth is a wireless RF technology, which was initially intended for replacing cables. With such an intent in mind, Ericsson, Intel, IBM, Toshiba and Nokia form a Special Interest Group (SIG), in 1998. Soon after, the potential of a small size, low power and low cost wireless technology beyond cable replacement is realized. Version 1.0 of Bluetooth specification is released in July 1999 and version 1.0B is released by the end of the same year. Also in 1999, the number of promoter members of Bluetooth SIG is increased to nine with addition of four new members: 3COM, Agere (Lucent), Microsoft and Motorola. First Bluetooth products appear in Autumn 2000 and with the necessity of clarification to achieve full interoperability among Bluetooth devices, version 1.1 of the specification is released in February 2001. At the time being, majority of the Bluetooth products is version 1.1 compliant.

2.1.1 Basics

Bluetooth operates on 2.4 GHz ISM (Industrial, Scientific and Medical) Band, which is originally reserved for non-commercial use for industrial, scientific and medical purposes, internationally. Worldwide availability and license-free property of ISM Band makes it a clever choice for Bluetooth technology, which aims low price and global interoperability as two of its primary goals. Frequency Hopping Spread Spectrum (FHSS) technique is used to provide immunity against interference sourced from other devices operating on the same radio spectrum, which is one of the drawbacks of the ISM Band. Due to the Bluetooth's FHSS scheme, a Bluetooth device hops through 79 channels in a pseudo random manner at a rate of 1600 hops per second. On the other hand, to support another important goal of Bluetooth, which is minimal energy consumption, it is designed to have relatively short range (10 meters commonly, 100 meters at most) and low data rate (1 Mbps) compared to most of the other wireless products on the market such as IrDA (for data rate) [8] and IEEE 802.11 (both for data rate and communication range) [7].

Bluetooth devices can form *piconet*s, which are small, star-shaped Bluetooth networks, and a device in a piconet can communicate with other members of the piconet. A Bluetooth piconet typically has a master node and one to seven slave nodes, which have direct radio connection with the master. Figure 2.1 (a) shows a very simple piconet with one master and one slave, whereas Figure 2.1 (b) depicts a relatively complex piconet consisting of a master and five slaves. It is the master node that coordinates the traffic in a piconet. Master node polls slave nodes based on a Time Division Multiple Access (TDMA) scheme. At each time slot, the master and the slaves hop to a different channel and stay there until the next time slot due to FHSS technique used in Bluetooth. Since Bluetooth devices hop 1600 times per second, a time slot is $1/1600 = 625\ \mu$s long. Each piconet has a unique hopping sequence allowing coexistence of different piconets in the same vicinity.

Small size of piconets, both in coverage area and in number of participants, arises need for larger Bluetooth networks. Fortunately, Bluetooth specifications define such networks called *scatternet*s. Scatternets are actually piconets with overlapping coverage areas and common nodes at such regions, called *bridge*s, which are used to forward data between such piconets. A simple scatternet is depicted in Figure 2.1 (c). Although piconet construction is explained in detail in Bluetooth specifications, only the basic concepts about scatternet are given leaving issues such as construction, routing and scheduling as open problems.

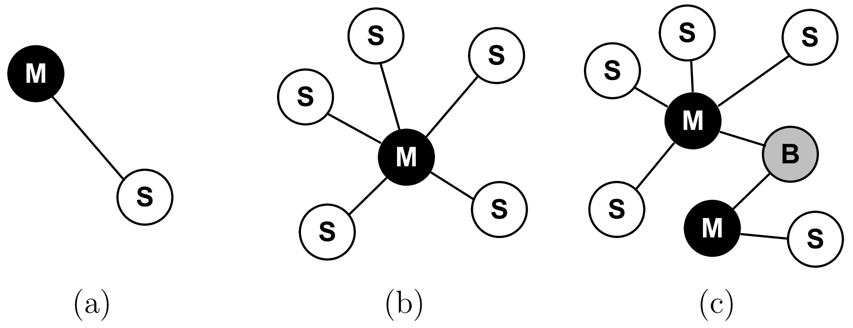


Figure 2.1: Various Bluetooth networks: (a) and (b) piconets, (c) a scatternet

The following sections contain more detail about Bluetooth technology. The discussions are held mainly around the protocol stack of Bluetooth.

2.1.2 Protocol Stack

Similar to other communication standards, Bluetooth contains certain protocols to accomplish its tasks and similar to other networking solutions these protocols are combined in a layered architecture. The Bluetooth protocol stack is shown in Figure 2.2. Protocols constituting the stack can be grouped into two as the *transport* and the *middleware* protocols. The transport protocols consist of *Radio* and *Baseband* protocols, *Link Manager Protocol* and *Logical Link Control and Adaptation Protocol*. *Host Controller Interface* can also be examined in transfer protocols, although it is not a protocol by itself and is just an interface as its name implies. On the other hand, *Service Discovery Protocol*, *RFCOMM* and *Telephony Control Signaling Protocol* can be listed as Bluetooth specific middleware protocols along with adopted protocols such as *Point-to-Point Protocol*, *TCP-UDP/IP*, *Object Exchange Protocol*.

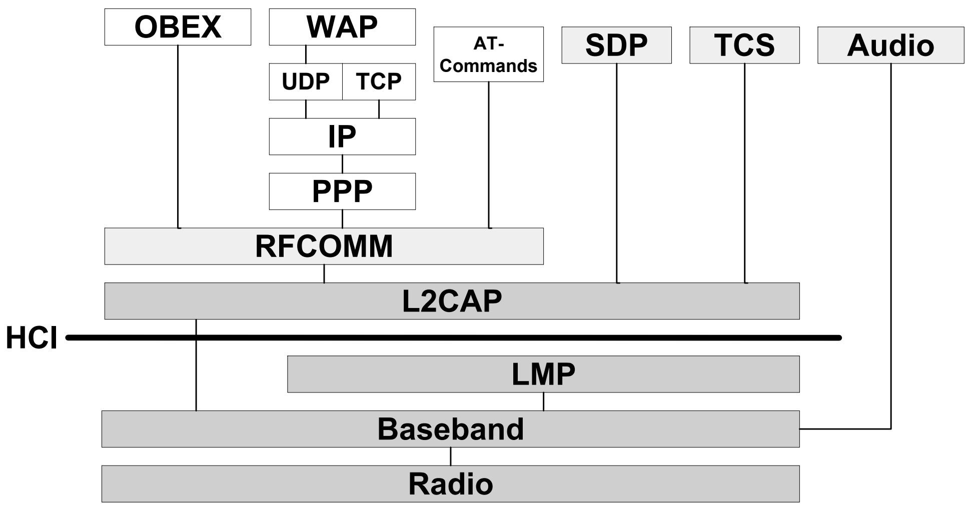


Figure 2.2: Bluetooth protocol stack

2.1.2.1 Transport Protocols

The Radio

This layer defines Bluetooth's air interface. It operates on ISM Band as mentioned before. More specifically, it operates between 2.402 GHz and 2.480 GHz and applies a fast (1600 hops/second) FHSS technique within this radio spectrum. A Bluetooth device hops through 79 channels, which are displaced by 1 MHz, in a pseudo random fashion. The basic formula for the frequency, **f**, of a Bluetooth device for a given timeslot t is like $\mathbf{f} = 2402\text{MHz} + f(t) \times 1\text{MHz}$ where $f(t) = 0, 1, 2, ..., 78$.

In order to reduce the transceiver complexity, a Gaussian shaped binary frequency shift-keying (GFSK) is used as the modulation technique. The transceiver operates at a rate of 1 Msymbols/sec, which is equal to a raw data rate of 1 Mbits/sec.

Although nominal link range of Bluetooth radio is between 10 centimeters and 10 meters, up to 100 meters of range can be achieved by increasing the transmit power. Bluetooth specifications define three power classes based on the transmit power to meet the requirements of devices with different purposes: Class 1 (20 dBm, 100 mW), Class 2 (4 dBm, 2.5 mW) and Class 3 (0 dBm, 1 mW). Furthermore, a Bluetooth device can adjust its transmit power as necessary. Since power adjustment is mandatory above 4 dBm of transmit power, Class 1 devices needs to implement it, whereas this is optional for Class 2 and Class 3 devices.

The Baseband

Basics of device communication using Bluetooth technology are described in this layer. Piconets, their creation, Bluetooth links and low-level packet types are defined by the baseband. Baseband layer is also responsible for sharing of the transmit resources within a piconet.

Bluetooth devices have two important parameters in the sense that they are involved in every aspect of Bluetooth communication: Bluetooth device address (BD_ADDR) and clock. Each Bluetooth device has a unique and permanent 48-bit address identifying it, which is assigned at manufacture time. Furthermore, each Bluetooth device is equipped with a free running 28-bit clock that ticks once every 312.5 μs, which is half of a Bluetooth time slot length. These two parameters are required to initiate communication between two Bluetooth devices.

Piconets - As previously mentioned, Bluetooth devices form piconets consisting of a master and at most seven active slaves. Piconet construction takes place in an ad-hoc manner. It should be noted that master/slave roles are not assigned at manufacture time, instead, determined during piconet construction process, which means that, each Bluetooth device is capable of being both master and slave. It should also be noted that the master/slave roles are with respect to a particular piconet, that is, a Bluetooth device can have different roles at different piconets simultaneously.

Within a piconet, each slave is assigned with a 3-bit active member address (AM_ADDR). Master coordinates the communication within the piconet as described previously. Although, there can be at most seven *active* slaves in a piconet, the maximum number of the member nodes of a piconet is actually far beyond this number. Such devices, which are not active but still registered with the master, are called to be in *parked* mode. Bluetooth devices, which have no association with any piconet, are called to be in *stand-by* node. Bluetooth device modes are discussed in more detail in next sections.

Members of a piconet follow the same hopping sequence, which is determined by the Bluetooth device address (i.e. BD_ADDR) and clock of the master, in a synchronized manner. Transmission and reception time axis are slotted, each slot having a length of 625 μs, which is equal to the duration between two frequency hops. Normally, a baseband transmission resides within boundaries of a slot, but three- and five-slot packets are also allowed. Hence, a single packet transmission may occupy one, three or five slots in which the transmit frequency does not change (i.e. transmitting and receiving devices do not hop during the transmission of multi-slot packets). At the end of a multi-hop packet transmission, the hopping sequence resumes with the frequency that would have been used if the normal hopping sequence were followed. Slaves maintain a clock offset, which is the difference between their Bluetooth clock and that of their master, in order to synchronize themselves with the slots determined by the master. Slots used for data transmission and reception are identified as odd and even according to the second least significant bit of the Bluetooth clock of the master. Even numbered slots are reserved for the transmission of master node, whereas at an odd numbered slot one of the slave nodes transmits. Hence, the master and slave nodes alternate transmission opportunities in a Time Division Duplex (TDD) manner. Furthermore, a slave can transmit only if master has sent data to it in previous slot.

As mentioned earlier, Bluetooth has a well-defied specification for piconet construction, which is summarized herein. Bluetooth connection establishment is a two-step procedure, where the first step is optional and is not executed if the node, which wants to initiate the communication, knows the BD_ADDR of the other node. BD_ADDR is obtained in the inquiry step if it is unknown.

The inquiry step is actually the device discovery process in which the candidate master of a potential piconet discovers the other devices in the vicinity. The candidate master advertises its presence with inquiry messages. Devices, which are performing inquiry scan meanwhile, respond with inquiry response messages, which includes the BD_ADDR and Bluetooth clock value of the device. The inquiry process is carried on a well-defined frequency hopping sequence with a period of 32 hops. The inquiry scanning device listens one of the 32 channels used during inquiry process for 10 ms long once every 1.28 seconds and the inquiring device transmits at two successive channels within a regular slot time. Once the inquiry process is accomplished, the devices have BD_ADDR and Bluetooth clock values of each other and ready for the page process in which the connection is actually established. Page step is carried very similar to inquiry step except that, this time, the hopping sequence of transmitting and receiving devices is determined by the BD_ADDR of the node previously inquiring (i.e. candidate master). Upon reception of a page message, the page scanning device responds with page response message, which leads to connected state for slave after reception of frequency hop sequence (FHS) packet, that is described soon, from master and for master after reception of acknowledgement of the FHS packet from the slave. Steps of paging procedure is shown in Figure 2.3.

Link and Baseband Packet Types - Once the connection is established, data can be exchanged between the nodes. Two types of links are supported in Bluetooth for data exchange: Asynchronous connectionless (ACL) link, and synchronous connection-oriented (SCO) link. ACL link is designed for asynchronous data traffic, which works in a best effort fashion whereas a SCO link supports synchronous 64 kbps communication in both directions and used especially for audio transmission.

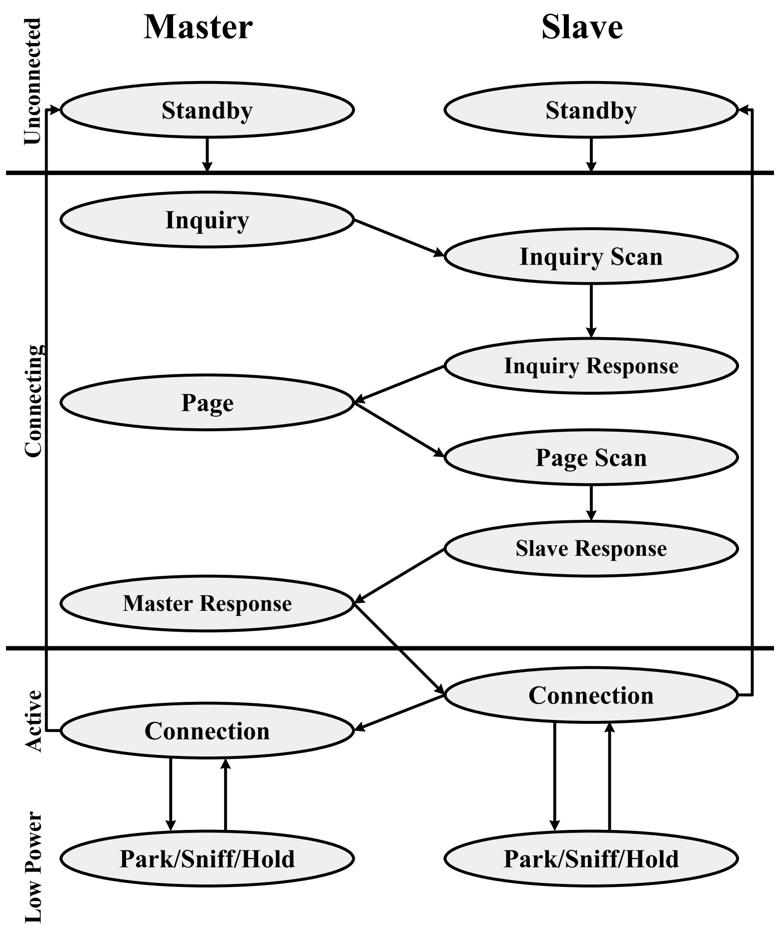


Figure 2.3: Bluetooth connection establishment

Various Baseband packet types are depicted in Figure 2.4. All packet types contain an access code (AC) field, which is used to distinguish packet transmissions of different piconets. All, but ID, packet types have a header field. All, except ID, poll and null packet types carry payload.

The poll packet is used by a master node when it has no data to send to a slave but still wants to poll it. The null packet is used to acknowledge a transmission without sending payload information. The frequency hope sequence packet is used to exchange BD_ADDR, AM_ADDR and Bluetooth clock information between potential master and slave nodes during paging process.

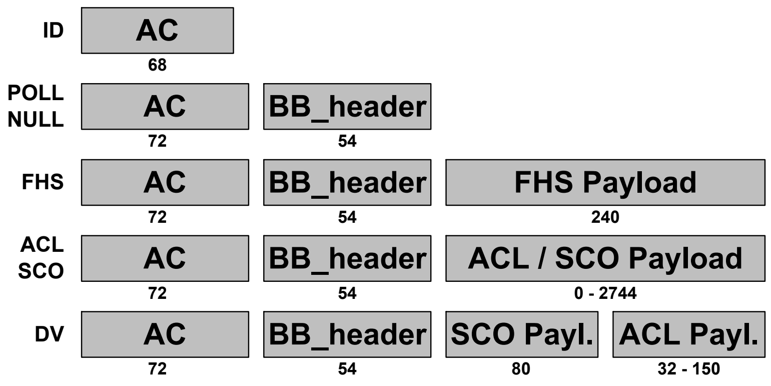


Figure 2.4: Baseband packets

The ACL and SCO packets are used for carrying asynchronous and synchronous data respectively. Asynchronous property of the ACL packets is due to presence of different packet sizes (i.e. single-, 3- or 5-slot packets). Most common ACL packets are DM1, DH1, DM3, DH3, DM5 and DH5. D stands for data whereas M and H stand for medium and high respectively to represent either Data - Medium rate or Data - High rate. Numerals represent the number of slots that the packets cover. Addition to information data, the payload includes 16-bit Cyclic Redundancy Check (CRC) code. ACL packets can be 2/3 FEC encoded to recover from errors encountered to some extent. ACL packets listed by now are retransmitted if they are either lost or corrupted, using an Automatic Repeat Request (ARQ) scheme. Last type of ACL packet is AUX1, which has no CRC code and is not retransmitted. SCO packets have no CRC and are not retransmitted similar to AUX1 packet but can be protected with a 1/3 or 2/3 FEC scheme if desired. SCO packet types are named as HV1, HV2 and HV3, which have 1/3, 2/3 and none FEC encoding respectively. To get a constant communication rate of 64 bps for an SCO link, SCO packets contain either 10, 20 or 30 bytes of user payloads depending on FEC scheme they apply. Data voice (DV) is the last type of baseband packet, which is a combination of ACL and SCO packet types. Details of AC, baseband header, SCO payload and ACL payload are depicted in Figure 2.5 for further examination.

Rates achieved by combination of different ACL packet types are depicted

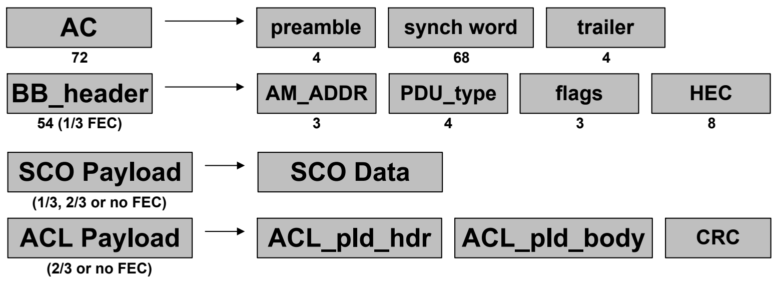


Figure 2.5: Baseband packet fields

in Table 2.1. It can be noticed that as the length of packets increase the communication rates also increase since the header overhead per payload and time wasted during switching to different channel decrease. The maximum rate that a Bluetooth link can achieve on single direction is 723 Kbps, whereas the maximum rate for a symmetric communication is 868 Kbps in total.

Table 2.1: Communication rates for different ACL packet types

Types	Payload Header (bytes)	User Payload (bytes)	FEC	CRC	Symmetric Max. Rate (kbps)	Asymmetric Max. Rate Forward (kbps)	Asymmetric Max. Rate Reverse (kbps)
DM1	1	0-17	2/3	yes	108.8	108.8	108.8
DH1	1	0-27	no	yes	172.8	172.8	172.8
DM3	2	0-121	2/3	yes	258.1	387.2	54.4
DH3	2	0-183	no	yes	390.4	585.6	86.4
DM5	2	0-224	2/3	yes	286.7	477.8	36.3
DH5	2	0-339	no	yes	433.9	723.2	57.6
AUX1	1	0-29	no	no	185.6	185.6	185.6

The Link Manager Protocol (LMP)

This protocol layer is primarily responsible for link set-up between Bluetooth devices. Authentication of Bluetooth devices is done by LMP using a challenge-response scheme. Encryption can be used for subsequent communication if desired once the authentication is succeeded. During the set-up phase of a Bluetooth link, information necessary for the rest of the communication such as supported links, packet types and power consumption modes are exchanged.

There are three types of low power modes, which are *sniff*, *hold* and *park*, that a Bluetooth device can use either to save energy during low activity periods or suspend its communication in a piconet and do other jobs such as participating another piconet, scanning, paging, inquiring. Although the low power modes are part of baseband, since they are managed by LMP, they are discussed in this section. In sniff mode, slave and master nodes agree on certain slots that a slave is going to listen the transmission of its master. In hold mode, a communicating pair agrees on a certain period of time that they will not communicate for. As the last type of power saving modes, in park mode, a slave node agrees with its master to suspend its active participation in the piconet until further notice. However, a parked slave continues to listen the master node at certain transmission slots called beacon instant. Unlike sniff and hold modes, the slave node releases its AM_ADDR in park mode and it is assigned a 8-bit parked member address (PM_ADDR), which is used for *unpark*ing, is assigned to it.

The Logical Link Control and Adaptation Protocol (L2CAP)

L2CAP layer hides the details of lower layers from the higher layers and provides a packet interface to them. For example, master/slave notions or changes in the frequency are unknown to the layers above L2CAP. Multiplexing is one of the main tasks of this layer along with segmentation and reassembly. Multiplexing is required since several protocols such as SDP, RFCOMM and TCS Binary run over it. Large packet sizes arriving from higher layers are divided into smaller packets to fit into maximum transferable unit (MTU) size of lower layers (i.e. segmented) and small packets arriving from lower layers are combined to construct the original packets (i.e. reassembled) at this layer. Providing Quality of Service (QoS) is also responsibility of L2CAP. Hence, QoS parameters are exchanged during establishment of an L2CAP link and used as necessary.

Host Control Interface (HCI)

This layer is designed to provide a uniform interface for accessing Bluetooth hardware capabilities. Using the HCI a host can pass data destined to or receive data coming from another Bluetooth device. Through the HCI, a host can also instruct the baseband to create a link with a specific device, initiate inquiries, request change of power save mode, etc. Although HCI is not a protocol layer, it is as important since it provides standardization at the layer that the host accesses the features of Bluetooth.

2.1.2.2 Middleware Protocols

Middleware protocols are intended to provide connection between transport layer protocols and Bluetooth aware applications. Unlike transport layer protocols, which are involved in every communication, not every middleware protocol takes part in a Bluetooth communication all the time. Bluetooth specific middleware protocols are RFCOMM, Service Discovery Protocol (SDP) and the Telephony Control Signaling (TCS) protocol, which are discussed shortly in this section. Some already existent protocols that are adopted to use with Bluetooth are also mentioned here.

RFCOMM emulates the RS-232 standard, hence, provides a serial interface to the packet based Bluetooth transport layers. There are many applications that use the traditional serial interface for communication which makes RFCOMM an important part of Bluetooth specifications since it enables a virtual serial port for such legacy applications.

In order to support a large variety of applications in the dynamic environment of Bluetooth devices SDP is developed, which enables to query what kind of services are available at another Bluetooth device. Although SDP provides means of learning services of another device across a Bluetooth link and how to get access to them, it does not provide access. Once the necessary information about the services is gathered via SDP, normal piconet operations can be used to access them.

TCS defines call control signaling to establish speech and data call between Bluetooth devices. Two ways to accomplish this task is designed. One of them is based on old AT command set, which is referred as TCS-AT. Since AT commands go through serial lines, TCS-AT makes use of RFCOMM. Bluetooth specifications also define a telephony control signaling protocol, called TCS-BIN, which is packet oriented. Different from TCS-AT, which only supports point-to-point configuration, TCS-BIN supports both point-to-point and point-to-multipoint configuration.

In order to provide interoperability with large set of applications that make use of protocols that were available before Bluetooth, such common protocols are adopted to use with Bluetooth. Following protocols can be used as parts of Bluetooth protocol stack: Point-to-point protocol (PPP), object exchange protocol (OBEX), infrared mobile communications (IrMC), TCP/UDP/IP, WAP, etc.

2.1.3 Profiles

Bluetooth specifications not only contain communication protocols but also some application level definitions, which are actually *profiles* of the Bluetooth. Profiles specify how the interoperable solutions can be developed for common usage scenarios. Besides supporting development of interoperable applications, profiles are also important since they provide some basic and useful, ready to use applications for consumers.

Figure 2.6 depicts profiles defined in Bluetooth specification, version 1.1. As it can be seen from the figure, some profiles depend on other profiles. Starting from the general profiles, which are Generic Access Profile, Service Discovery Application Profile, Serial Port Profile and Generic Object Exchange Profile, profile definitions in Bluetooth are described shortly in the rest of this section.

General Access Profile, GAP - Bluetooth device discovery and connection establishment is defined in GAP. Security management is also defined in this profile. Since GAP is a generic profile, other profiles and applications can refer to it for the tasks mentioned.

Service Discovery Application Profile, SDAP - SDAP defines investigation of services provided to a Bluetooth device. SDAP is built on top of GAP and makes use of SDP.

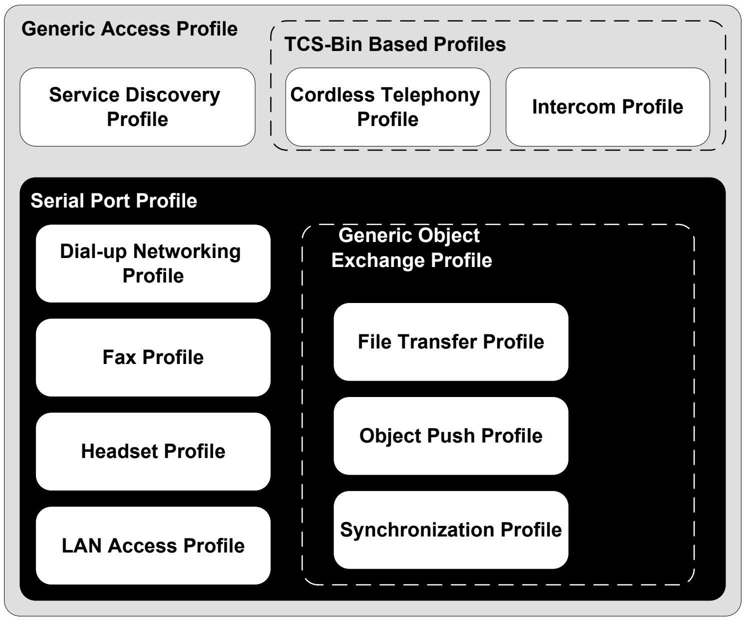


Figure 2.6: The Bluetooth profiles

Serial Port Profile - This profile defines requirements of Bluetooth devices in order to setup emulated serial cable connection using RFCOMM. Similar to SDAP, Serial Port Profile depends on GAP.

Generic Object Exchange Profile, GOEP - This profile defines how applications can support object exchanges. It depends on Serial Port Profile.

Cordless Telephony Profile - defines means of using a telephone connected to a base station of a fixed telephony network via Bluetooth.

Dial-Up Networking Profile - defines means of using a nearby telephone or modem as a wireless modem to access internet or other dial-up services.

Fax Profile - defines how a Bluetooth enabled device use a Bluetooth phone or modem as a wireless fax modem to send and receive fax messages.

Headset Profile - defines the requirements to support wireless headset usage where the communication link is Bluetooth.

LAN Access Profile - defines means of accessing services of a local area network using PPP over RFCOMM.

File Transfer Profile - defines how a Bluetooth device browses and edits objects (i.e. files and folders) in the file system of another Bluetooth device.

Object Push Profile - defines methods to push, pull or exchange simple objects such as business cards between two Bluetooth devices.

Synchronization Profile - defines means of data synchronization of two Bluetooth devices when they share a common coverage area.

Additional to the profiles shortly described above, some other profiles have been introduced later. They are listed below just to mention their names:

- Generic Audio/Video Distribution Profile (GAVDP)
- Advanced Audio Distribution Profile (A2DP)
- Audio/Video Remote Control Profile (AVRCP)
- Basic Imaging Profile (BIP)
- Basic Printing Profile (BPP)
- Hardcopy Cable Replacement Profile (HCRP)
- Bluetooth Extended Service Discovery Profile (ESDP) for Universal Plug and Play™(UPnP™)
- Hands-Free Profile (HFP)
- Human Interface Device Profile (HID)
- Common ISDN Access Profile
- Personal Area Networking Profile (PAN)
- SIM Access Profile (SAP)

Chapter 3

Related Work

This chapter gives information about previous studies related to the study presented in this thesis. First, some criteria to classify scatternet construction algorithms are presented and then some influential algorithms in the literature proposed for scatternet construction problem are discussed. Algorithms aiming bandwidth-efficiency and studies examining relations between scatternet topology and bandwidth-efficiency are also mention in this chapter.

There have been many solution proposals for the Bluetooth scatternet construction problem. There are several ways to classify them. A scatternet construction algorithm can be classified as central or distributed, single-hop or multi-hop, static or dynamic as described in [6]. In central algorithms, a distinguished node decides on the scatternet topology whereas in distributed ones nodes participate to the construction process independently with local information they have. Single-hop algorithms work only if all the nodes are in the communication range of each other. Although, multi-hop algorithms do not have such a constraint, for the sake of connectedness some nodes should share some common coverage area. Dynamic algorithms can handle arriving and leaving nodes, while in static algorithms the nodes should start the construction process at the same time and the set of nodes constituting the scatternet can not be changed once the scatternet is constructed. Another classification criterion for scatternet construction algorithms is according to their primary focus, which may highly vary.

Many of the proposals primarily pay attention to the efficiency of the scatternet construction algorithm, considering the duration of the construction process and number of messages exchanged during construction as it is the case in [20], [13] and [14], all of which are single-hop and static algorithms. The algorithm proposed in [20] consists of coordinator election and role determination phases and achieves a fully connected topology with minimum number of piconets.

In [22] and [27] tree-shaped topologies are formed for the ease of routing as another primary focus, where the former is a static algorithm and the latter is a dynamic one. Another construction algorithm resulting with a tree-shaped topology is [23], which is a decentralized, dynamic and single-hop algorithm, having the same motivation as the algorithms in [22] and [27], that is easy routing. The algorithm proposed in [23] is especially designed for the environments in which handling arriving and leaving nodes is more important than high throughput.

A state-of-art algorithm, since it is multi-hop, dynamic and distributed, is proposed by J. Yun, J. Kim, Y.-S. Kim, and J. Ma [26]. It is a three-phase algorithm. In the first phase neighbors are discovered, in the second phase neighbors are grouped and in the third phase roles are assigned to the nodes to establish a link with one node in each group determined in the second phase. The resulting topology is between two extremes, topologies in which every possible link is established being on the one extreme and tree shaped topology being on the other extreme. Another multi-hop, dynamic and distributed scatternet construction algorithm is proposed by J. Ghosh, V. Kumar, X. Wang and C. Qiao [6], in which “spin” is defined as the period that the device is in inquiry and inquiry scan modes. A node “spins” at certain intervals for device discovery and the establishes new connections according to the rules defined in [6].

Other than these, some of the algorithms apply heuristics in order to make the constructed scatternet efficient in terms of some metrics. One such heuristic is to keep the number of piconets as small as possible in the resulting scatternet with the goal of decreasing the amount of inter-piconet traffic and interference, as it is the case in [20]. However, as studied in [28], there is not significant amount of interference between piconets that are using FHSS technique with different pseudo-random frequencies to cause a substantial amount of performance degradation, unless the number of overlapping piconets exceeds sixty. Besides, increasing the number of piconets may result in a higher total scatternet capacity and lower intra-piconet traffic. There exist other studies, which focus on different aspects of the problem. Some studies like [19], on the other hand, aim to form fault-tolerant topologies by means of constructing multiple (i.e. alternative) paths between nodes.

H. Sreenivas and H. Ali [21] propose a genetic algorithm to determine master, slave and bridge roles in the scatternet. On the other hand, in the algorithm proposed by Y. Kawamoto, V.W.S. Wong and V.C.M. Leung [12], a control scatternet is formed to accomplish the tasks about control issues of scatternet such as dynamic join and leave of nodes and route discovery. The actual data is transferred through links that are established on demand and destroyed when the data transfer is over.

Although various proposals have been made to form scatternets with various objectives and considering various factors, algorithms that favor efficient usage of bandwidth are not widely studied. One of the earliest study in this topic is by D. Miorandi and A. Zanella [18], which is actually about piconets rather than scatternets. In this study, traffic patterns among the piconet members is taken into consideration while selecting the master unit of the piconet. Some studies on constructing efficient scatternet topologies are [2], [15] and [24]. All three proposals are single-hop, central and static solutions. Baatz et al. [2] propose a graph theoretical solution which uses 1-factors to construct a scatternet. Point of view of algorithm proposed in [15] is a min-max optimization. On the other hand, T. Topal [24] uses a set of heuristics to determine the locations and the roles of the nodes, assuming that the traffic flow information is known a priori. Along with the scatternet construction proposals considering the relation between topology and efficiency, there are studies that try to reveal the relation between them such as [16], [10] and [17]. R. Kapoor, M. Sanadidi and M. Gerla [10] apply an analytical approach where D. Miorandi, A. Trainito and A. Zanella [17] try to establish a mathematical background to determine the relations between the topology and network capacity. On the other hand, Miklos et al. [16] apply a statistical approach in order to investigate the effects of topology on performance.

In this work, we propose a distributed algorithm that constructs and maintains a scatternet topology to make it bandwidth-efficient considering the traffic demands among nodes of the scatternet. The maintenance part of the algorithm is triggered when there is a substantial change in the amount of traffic flow between nodes, and therefore the algorithm is adaptive to the traffic dynamics and preserves the bandwidth-efficiency in changing traffic conditions. The fundamental approach that we use in our algorithm is about bringing the nodes that are heavily communicating with each other closer in the scatternet. This is also the approach followed by [24]. As we have discussed earlier, this approach leads to better utilization of scatternet capacity, and decreases the average end-to-end delays of packets and total energy consumption of nodes.

Chapter 4

Scatternet Construction Algorithm

This chapter contains details of the algorithm proposed for construction and maintenance of bandwidth-efficient scatternets. First, an overview of the algorithm is presented and then operations constituting procedures are discussed. Finally, how several building blocks come together to construct procedures constituting the algorithm is explained.

4.1 Algorithm Overview

The algorithm we propose produces a scatternet topology in which the available bandwidth is efficiently used as its primary goal. The algorithm works in distributed fashion and is adaptive to both joining/leaving nodes and changes in the traffic pattern. For a specific set of nodes, if further modifications do not lead to any improvements, the scatternet topology is said to become stabilized and it remains in the stable state until there is a change either in the traffic pattern, which makes the current topology bandwidth-inefficient, or in the set of nodes constituting the scatternet.

The algorithm basically consists of two *concurrent* main procedures: *Maintenance* and *Link Establishment*. The *Maintenance* procedure maintains the bandwidth-efficiency of a scatternet by modifying the current topology. On the other hand, the *Link Establishment* procedure handles new connections and constructs a base topology that the *Maintenance* procedure can work on.

In order to perform its primary goal, the *Maintenance* procedure constantly collects traffic flow information and runs a set of operations based on this information when "significant" changes in traffic patterns are observed. These operations are performed in order to modify the scatternet topology so that the nodes, whose communication demands are relatively higher, are placed closer to each other in the scatternet. The *Maintenance* procedure depends on some assumptions. The first assumption is that all the nodes, which are going to be part of the scatternet, are in the communication range of each other. This assumption is valid in many cases such as audience in a conference room, students in a classroom, etc. Although, the algorithm proposed here is based on this assumption, it can be modified to work also when this assumption is not the case, as it is discussed in Section 6.2. Another assumption is that the topology that the *Maintenance* procedure works on has the following properties:

- There does not exist any master/slave bridges
- Bridge nodes have only two master nodes

The *Link Establishment* procedure should take these restrictions into account, while establishing new links. The *Maintenance* procedure also preserves these properties while modifying the topology to achieve bandwidth-efficiency. Similar to the case in the first assumption, the restrictions on the properties of the topology can be eliminated with slight modifications on the operations of the *Maintenance* procedure. However, they are presented for the efficiency of the resulting scatternet topology. As studied in [9], master/slave bridges constitute a burden for the load balancing and simultaneous communication in different piconets. When a bridge master node switches to be a slave, all its slaves (the entire piconet) become useless and their resources are wasted. On the other hand, switching time of bridges cannot be underestimated and sharing a bridge's time for more than two piconets would make that bridge a bottleneck in the scatternet. We also assume that the underlying routing protocol uses the shortest paths between any two communicating nodes. This assumption is crucial in order to result with a bandwidth-efficient scatternet topology. The proposed algorithm can still work with a routing protocol that does not satisfy this condition, but such a routing protocol contradicts with the basic approach of the algorithm, which is discussed in Chapter 1. The cost metrics that we use in order to evaluate the efficiency of the scatternet topology are also based on this last assumption.

The *Maintenance* procedure is composed of following operations, which are combined in a certain way to obtain bandwidth-efficiency:

- **Master/Bridge Reassignment:** Reassigns master and bridge roles to the nodes in a way that the bandwidth usage within the piconet is minimized.
- **Piconet Division:** Splits a piconet into two when the bandwidth usage within the piconet exceeds a threshold.
- **Slave Transfer:** Transfers one or more slave nodes from their current piconet, to neighboring piconets.
- **Piconet Merge:** Merges two piconets into one.

Although all these operations make local modifications spanning single piconet or two piconets, by taking place concurrently or one after the other at different locations of the scatternet, they lead to global adaptation of the scatternet topology to the traffic flow pattern.

In order to change the scatternet topology according to the traffic flow patterns to get a bandwidth-efficient scatternet, this information should be acquired somehow. The method to estimate the flow patterns is explained in the next section. Section 4.3 describes how the operations use the traffic flow information, once it is acquired and Section 4.4 describes how the operations are combined to construct the *Maintenance* procedure, which is the heart of the algorithm for achieving bandwidth-efficiency, by a fusion algorithm. Section 4.5 describes the details of the *Link Establishment* procedure, which handles new connections in a way that a base topology that the *Maintenance* procedure can work on is generated.

4.2 Collecting Traffic Information and Estimating Traffic Flow Rates

Since the scatternet topology is modified according to traffic flow patterns, which are due to communication demands between nodes in the scatternet, traffic flow information is required by the algorithm and must be collected to estimate the traffic flow rates. As mentioned before, all the data traffic within a piconet flows through the master node, which makes it the natural choice for the point that the traffic information is collected. The information collected from traffic that flows through the master node satisfies the majority of the information requirements of the operations used by the algorithm, but not the all. Hence, extra messaging required to collect information about traffic flows is minimized, but still needed due to the reasons explained shortly.

There exist three types of traffic in a piconet about which traffic flow information is gathered:

- **Intra-piconet:** Traffic between members of the piconet (i.e. Master-Slave and Slave-Slave communication).
- **Incoming/Outgoing:** Traffic flowing from/to members of the piconet to/from neighboring piconets.
- **Relayed:** Traffic forwarded to a neighboring piconet, which is received from another neighboring piconet.

Consider the piconet P_1 shown in Figure 4.1. It consists of six members: A master node, two slave/slave bridges S_0 and S_2 connected to neighboring piconets

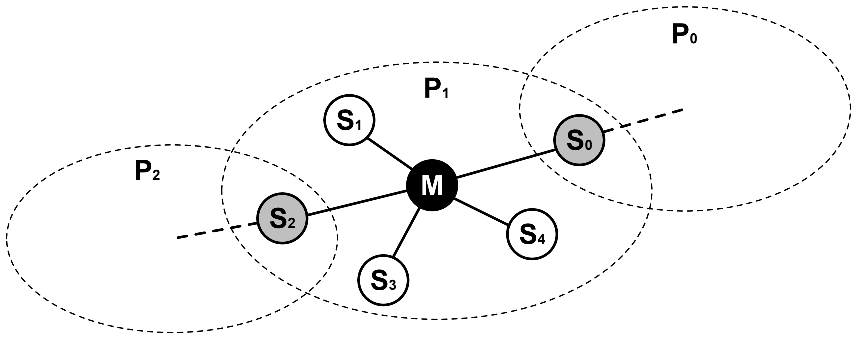


Figure 4.1: A sample piconet on which traffic data is gathered

P_0 and P_2 respectively, and three slave nodes S_1, S_3 and S_4. For such a piconet, a table, namely traffic table, shown in Figure 4.2 is constructed at the master node M.

	M	**S₀**	**S₁**	**S₂**	**S₃**	**S₄**	**P₀**	**P₂**
M								
S₀	░							
S₁	░	░						
S₂	░	░	░					
S₃	░	░	░	░				
S₄	░	░	░	░	░			
P₀	▒	▒	▒	▒	▒	▒		
P₂	▒	▒	▒	▒	▒	▒	■	

Figure 4.2: Traffic Table

Each cell in table shown in Figure 4.2 represents the amount of traffic flow rate observed between the entities that match corresponding row and column. As indicated with shaded cells in the table, only half of it is used since two-way traffic is not distinguished. In addition, three types of traffic that are previously introduced are indicated with different shadings.

As pointed out before, traffic flow inside a piconet passes through the master node and therefore, almost all information required to construct the traffic table is available without extra messaging. Message exchange is only needed for traffic flow between bridge nodes and corresponding neighboring piconets. For the scatternet shown in Figure 4.1, for instance, S_0 should send traffic flow rate between itself and piconet P_0 to the master node, M, at certain intervals. A similar messaging is required for the traffic flow between S_2 and P_2.

The rate of traffic flow may change rapidly, which would lead to constant alteration of the scatternet topology. In order to prevent that, aging method can be used while collecting the traffic data to estimate the traffic patterns, which is proposed in [24] and presented in Equation 4.1.

$$avgRate_t = \alpha \times avgRate_{t-1} + (1 - \alpha) \times instRate_t \tag{4.1}$$

This method smoothly integrates changes in the instantaneous traffic to the average traffic rate and its effect can be adjusted by varying α between 0 and 1. As α gets closer to 1, effect of recent traffic flow information on average traffic flow patterns increases.

Please note that, as new nodes join or existing nodes leave the structure of the table should be changed. When a new node joins or an existing slave connects to another master resulting a new neighboring piconet, corresponding rows and columns should be added to the traffic table. Similarly, when a node leaves the piconet or one of the bridges loses its connection to its other master, corresponding rows and columns should be deleted from the table. Once the new row and column is added to the table, monitoring related to the new entry begins.

4.3 Operations

Before explaining details of each operation, giving a classification of them may be helpful to understand them better. There are two ways to distinguish and classify these operations. In terms of scope, the *Master/Bridge Reassignment* and the *Piconet Division* operations take place in a single piconet, although they may lead to changes on the interface of the piconet to its neighboring piconets (i.e. bridge nodes), and therefore named as intra-piconet operations. The *Slave Transfer* and the *Piconet Merge* operations, on the other hand, need coordination of two neighboring piconets, and named as inter-piconet operations. From another point of view, the *Master/Bridge Reassignment* operation only switches roles of the scatternet members and does not alter the overall scatternet topology. Other three operations (i.e. *Slave Transfer*, *Piconet Division*, and *Piconet Merge*) on the other hand, change the topology. In the following subsections, details of operations about how they use the traffic pattern information and how they affect the scatternet are given.

4.3.1 Master/Bridge Reassignment

This operation reassigns master and bridge roles to the members of the piconet so that the cost function introduced in Equation 4.2 is minimized.

$$C_p = C_{intra-p} + C_{inter-p} \tag{4.2}$$

$$C_{intra-p} = \sum_{s=0}^{n-1} \mathbf{T}_{S_s M} + \sum_{s_1=0}^{n-1} \sum_{s_2=s_1+1}^{n-1} 2 \times \mathbf{T}_{S_{s_1} S_{s_2}} \tag{4.3}$$

$$C_{inter-p} = \sum_{p=0}^{m-1} \mathbf{T}_{P_p M} + \sum_{p=0}^{m-1} \sum_{\substack{s=0 \wedge \\ s \neq bridge_p}}^{n-1} 2 \times \mathbf{T}_{P_p S_s} \tag{4.4}$$

The cost function is composed of two components: Cost of intra-piconet traffic (i.e. $C_{intra-p}$) and cost of inter-piconet traffic (i.e. $C_{inter-p}$). $\mathbf{T}$ represents the traffic table and subscripts identify row and column indexes. n is the number of slaves and m is the number of neighboring piconets. It can be noticed from the equation that traffic flow rates between communicating entities multiplied by the length of corresponding communication paths are summed up to find the cost of carrying traffic in the piconet. Since, all traffic flow passes through the master node, traffic flow rate between two slave nodes, and traffic flow rate between a slave node and a piconet is multiplied by two. There is no cost of communication between a bridge node and the corresponding neighboring piconet. Such costs are cancelled for $C_{inter-p}$ by skipping slave nodes that are also bridge nodes to the piconet of which traffic flow rate is counted ($bridge_p$). In addition, there exists a cost of relayed traffic. However, this cost is not affected by the selection of different master and bridge nodes. That is why it is not included in the cost function.

Algorithm 1 describes the execution steps of the *Master/Bridge Reassignment* operation. The nested loops generate all possible master and bridge assignments and for each such assignment the cost function, C_p, is calculated. Note that the assignments at Line 7 is required due to the bridge node check in Equation 4.4. If the current cost is the minimum among the ones calculated by now, current master/bridge configuration is saved. At Line 20, we have the master/bridge configuration that minimizes the bandwidth usage within the piconet and roles are reassigned to the nodes according to it. Since, there are n choices for the master node, and m out of $n-1$ nodes will be chosen as bridges and these bridges can be assigned to neighboring piconets in $m!$ ways, C_p is calculated $n!C(n-1,\ m)m!$ times.

If the master node changes because of the operation, traffic table is transferred to the new master node at which the *Maintenance* procedure will be executed next.

Although this operation rearranges roles inside a piconet, it has a global effect as illustrated in Figure 4.3. Suppose that nodes labeled as A and B heavily communicate with each other and they are initially located as in Figure 4.3 (a) in the scatternet. Master node of P_0 will assign node A as the bridge node for P_1, as in Figure 4.3 (b), since the cost function will be minimized in this way due to high traffic flow rate between node A and P_1. Afterwards, master node of P_1 will notice a high traffic flow rate between the new bridge node, A, and P_2. As it is depicted in Figure 4.3 (c), in result of the *Master/Bridge Reassignment* operation, node A will be the bridge node between piconets P_1 and P_2. After A becomes a member of piconet P_2 and master node of P_2 executes the *Master/Bridge Reassignment* operation, A will be the new master node in P_2

Algorithm 1 Master/Slave Reassignment

```
1: $min\text{-}C_p \leftarrow \infty$
2: $min\text{-}master \leftarrow$ NULL
3: $min\text{-}bridge_0 \leftarrow$ NULL, $min\text{-}bridge_1 \leftarrow$ NULL, ..., $min\text{-}bridge_m \leftarrow$ NULL
4: for all Nodes $M$ in the piconet do
5:   for all Permutations of neighboring piconets: $P_a P_b ... P_x$ do
6:     for all $m$ out of $n-1$ combinations of nodes, excl. $M$: $S_k S_l ... S_y$ do
7:       $bridge_a \leftarrow k$, $bridge_b \leftarrow l$, ..., $bridge_x \leftarrow y$
8:       Calculate $C_p$
9:       if $C_p < min\text{-}C_p$ then
10:        $min\text{-}C_p \leftarrow C_p$
11:        $min\text{-}master \leftarrow M$
12:        $min\text{-}bridge_0 \leftarrow P_a : S_k$
13:        $min\text{-}bridge_1 \leftarrow P_b : S_l$
14:        ...
15:        $min\text{-}bridge_m \leftarrow P_x : S_y$
16:      end if
17:    end for
18:  end for
19: end for
20: Rearrange master and bridge roles in the piconet according to $min\text{-}master$,
    $min\text{-}bridge_0$, $min\text{-}bridge_1$, ..., and $min\text{-}bridge_m$
```

to decrease the length of communication path with B to one, which is shown in Figure 4.3 (d).

It should be noted that, this is just a sample scenario. Also, it might have been the case that B approaches to A, or they could have met at P_1 depending on the order of execution of the operation by master nodes. However, at the end, nodes that heavily communicate with each other will be closer, as a result of successive execution of the *Master/Bridge Reassignment* operation in different piconets.

4.3.2 Piconet Division

The *Piconet Division* operation basically splits a piconet into two. A piconet is convenient for division, if the number of non-bridge slaves is at least two more

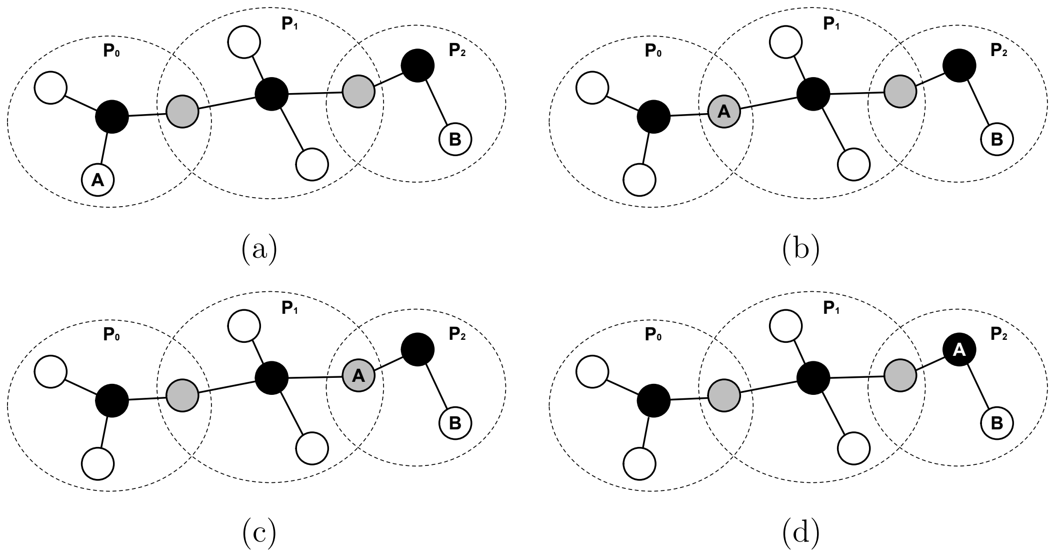


Figure 4.3: Global effect of the *Master/Bridge Reassignment* operation. (a) Initial topology. Topologies after execution of the operation at (b) P_0, (c) P_1 and (d) P_2, following this order.

than the number of neighboring piconets. Such a constraint is necessary because in the resulting topology, there will be one more master node and a bridge node that will connect two newly constructed piconets.

Similar to the *Master/Bridge Reassignment* operation, in the *Piconet Division* operation for all possible role assignment scenarios for the resulting topology, the result of cost function is calculated and the configuration minimizing the cost is chosen among others. On the other hand, the *Piconet Division* operation involves different set of roles than that of the *Master/Bridge Reassignment* operation. For example, opposed to one master role in the *Master/Bridge Reassignment* operation two master roles exist in the *Piconet Division* operation, since two piconets out of one is created. Similarly, other than assigning bridges to neighboring piconets, one more bridge is required between two new piconets of the resulting topology. Cost function in the *Piconet Division* operation is similar to equations 4.3 and 4.4 in the sense that it sums up traffic flow rates multiplied by corresponding communication paths for all communicating pairs. However, it differs from the cost function of the *Master/Bridge Reassignment* operation since the

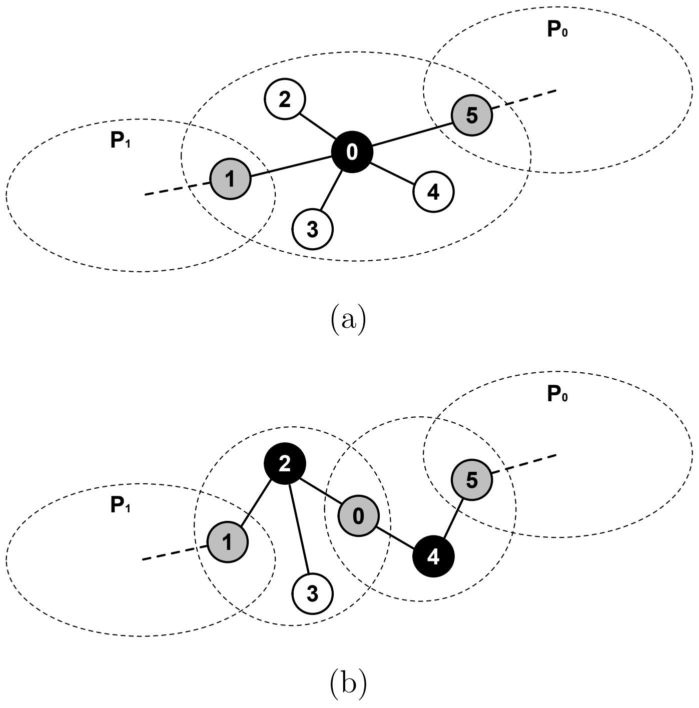


Figure 4.4: (a) Before and (b) after the *Piconet Division* operation

hop distances are either one or two in that operation, whereas resulting topology of the *Piconet Division* operation contains either one-, two-, three- or four-hop paths between the nodes, as it can be seen from Figure 4.4 (b).

Figure 4.4 depicts part of a scatternet topology before and after the *Piconet Division* operation. Once the split is accomplished, traffic table is also split into two according to distribution of nodes to piconets and two traffic tables are transferred to corresponding master nodes.

4.3.3 Slave Transfer

The *Slave Transfer* operation checks for each slave node in the piconet whether it is beneficial to assign the slave as a member of a neighboring piconet. If this is the case and if the neighboring piconet can accept a new slave node, such nodes are transferred to the corresponding piconets. Equations 4.5 and 4.6 shows how the gain, G_{ST}, and cost, C_{ST}, of a slave transfer is calculated. G_{ST} and C_{ST} are used to determine whether transferring a slave is beneficial as it will be explained shortly.

$$G_{ST} = \mathbf{T}_{S_{cs}P_{cp}} \tag{4.5}$$

$$C_{ST} = \sum_{\substack{p=0\wedge \\ p\neq cp}}^{m-1} \mathbf{T}_{S_{cs}P_p} + \sum_{\substack{s=0\wedge \\ s\neq cs\wedge \\ s\neq bridge_{cp}}}^{n-1} \mathbf{T}_{S_{cs}S_s} + \mathbf{T}_{S_{cs}M} \tag{4.6}$$

The gain and cost of the transfer is calculated for each non-bridge slave node, namely candidate slave node, denoted by *cs* and for each neighboring piconet that has less than eight members, namely candidate piconet, denoted by *cp*. The gain, G_{ST}, is the traffic flow rate between the candidate slave and candidate piconet. The cost of transferring a slave, C_{ST}, is the summation of traffic flow rates between the slave to be transferred and

- neighboring piconets other than the candidate piconet,
- other slave nodes of the piconet except the bridge node connected to the candidate piconet, that is $bridge_{cp}$, and
- master node.

In fact, G_{ST} and C_{ST} should be multiplied by two, because transfer of a slave node increments or decrements routing paths by two hops. Since this is the case for both cost and gain however, they are omitted.

Transfer of a candidate slave to a candidate piconet for which $(G_{ST} - C_{ST})$ is positive, is marked as a beneficial transfer. The greater this value is, the more the transfer is beneficial. After determination of beneficial transfers, each transfer is performed one by one starting from the most beneficial one down to the least beneficial one. It might be the case that a transfer cannot be performed because the candidate piconet happens to already have eight members. Such cases may occur when beneficial transfers with common target candidate piconets exist. In these cases, transfers with more benefit are performed, others are skipped.

Although this operation affects two piconets, successive execution of it has a global effect as depicted in Figure 4.5. Like the example scenario introduced for the *Master/Bridge Reassignment* operation, suppose nodes labeled A and B heavily communicate with each other and they are initially located as shown in Figure 4.5 (a). Master node of P_0 will notice that A communicates with P_1 more than it communicates with piconet members, and eventually it will transfer A to P_1, as depicted in Figure 4.5 (b). After that, P_1 will notice a high traffic flow rate between A and P_2. At the end, as the case in Figure 4.5 (b), A will be transferred to P_2, where B is also located.

Unlike the *Master/Bridge Reassignment* operation, the *Slave Transfer* operation does not change the roles (e.g. master, slave, bridge) of nodes, but modifies the scatternet topology by transferring nodes to different piconets.

4.3.4 Piconet Merge

The *Piconet Merge* operation can be thought as the reverse operation of the *Piconet Division* operation. Considering the example scenario previously given, the topology shown in Figure 4.4 (b) turns into the topology shown in Figure 4.4 (a) as a result of the execution of this operation. Algorithm 2 outlines the execution steps of the operation.

All neighboring piconets are tested for merge feasibility until one such piconet is found. If total number of nodes does not exceed eight and if total traffic flow rate does not exceed 1 Mbps, two piconets are merged.

As the first action, traffic tables are merged. Actually, this is not a straightforward task because traffic table of the resulting piconet cannot be constructed out of traffic tables of two merging piconets. As an example, in Figure 4.4 (b), traffic flow rate between nodes 1 and 5 cannot be obtained from traffic tables stored at nodes 2 and 4. Such information can be obtained by holding extra information

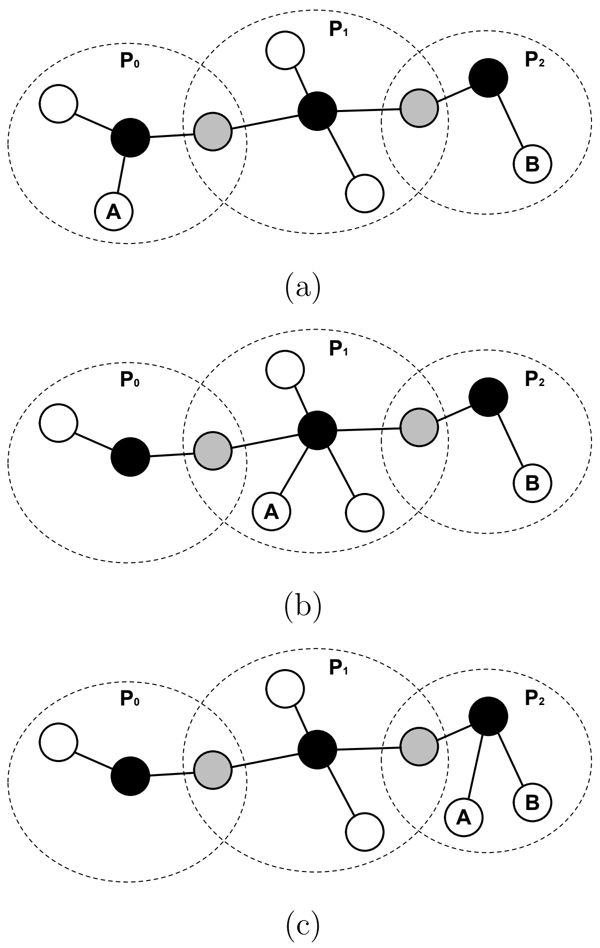


Figure 4.5: Global effect of the *Slave Transfer* operation. (a) Initial topology. Topologies after execution of the operation at (b) P_0 and (c) P_1, following this order.

and extra messaging. Instead, we apply a heuristic for completing the unknown parts of the merged traffic table. Consider two neighboring piconets P_0 and P_1 that are going to merge. From the traffic table of P_1, we know the amount of traffic flow streaming from P_1 to P_0. We do not know how much of it is indented for P_0 and how much of it is relayed. However, from the traffic table of P_0, we also know what percentage of traffic flow coming from P_1 goes to members of P_0. Combining these information gives us the amount of traffic flow streaming from P_1 to members of P_0, which are assumed to be equally shared by the piconet members.

Algorithm 2 Piconet Merge

```
1: for all Neighboring piconets $p$ do
2:     if (Total # of nodes in current piconet + Total # of nodes in $p$) $< 8$ then
3:         Merge traffic tables
4:         if Total traffic $< 1$ Mbps then
5:             Assign all nodes of $p$ as a slave to current piconet
6:             Execute Master/Bridge Reassignment operation
7:             Terminate
8:         end if
9:     end if
10: end for
```

After traffic tables are merged, total traffic flow rate is checked to see whether it exceeds 1 Mbps or not. If the 1 Mbps threshold is not exceeded, all nodes of the neighboring piconet are assigned as slave nodes to the master node running the *Piconet Merge* operation. The *Master/Bridge Reassignment* operation is run thereafter in order to assign roles in the newly constructed piconet according to the traffic table.

4.4 Maintenance Procedure

As indicated in previous sections, the *Maintenance* procedure is a combination of a set of operations. In this section, we present a fusion algorithm, which combines these operations and basically establishes an execution order.

Algorithm 3 The Fusion Algorithm

```
1: Execute Master/Bridge Reassignment operation
2: if Current node is still master then
3:     Execute Slave Transfer operation
4:     if Total piconet traffic $> 1$ Mbps then
5:         Execute Piconet Division operation
6:     else
7:         Execute Piconet Merge operation
8:     end if
9: end if
```

The fusion algorithm can be seen in Algorithm 3. This algorithm will be executed at master nodes, whenever a "significant" change is observed in the traffic table or total piconet traffic exceeds 1 Mbps. Such a bound is necessary because the capacity of a piconet is 1 Mbps. When traffic demand is more than that amount in a piconet, it should be split into two, in order to increase the capacity. When this is not the case, however, merging piconets as much as possible is beneficial for the sake of shortening the paths and decreasing the number of bridges that switch between two piconets.

Observation of significant change occurs when the value of $totalDiff$ calculated as in Equation 4.7 exceeds a threshold value. Adjustment of the threshold would affect the sensitivity of the *Maintenance* procedure to the changes in the traffic flow rate.

$$totalDiff = \sum_{i=0}^{t-1} \sum_{j=i+1}^{t-1} \left| T_{ij} - T_{last_{ij}} \right| \tag{4.7}$$

This equation gives the sum of differences between traffic flow rates for each pair of communicating entities (i.e. master node, slave nodes and piconets) represented in the current traffic table and the traffic table that was used in the last execution of the fusion algorithm. Initial execution of the algorithm is made after a timeout at which enough data will be present in the traffic table.

While the fusion algorithm is being executed on them, master nodes do not accept coordination for inter-piconet operations (i.e. *Slave Transfer*, *Piconet Merge*) of neighboring piconets.

4.5 Link Establishment Procedure

The main purpose of this procedure is to handle link establishments, which are either between nodes of the same scatternet, between nodes of different scatternets or between new nodes and the nodes of the scatternet. Although such a classification is given, the procedure does not make such a distinction and use one common approach for any link establishment.

The links are established after inquiry and page steps, as it is discussed in

Section 2.1.2.1. Inquiry and inquiry scan modes play a key role for device discovery and determination of master/slave roles. The *Link Establishment* procedure basically defines whether a node having a certain role (i.e. master, bridge, slave or free node) can switch to inquiry or inquiry scan modes. This definition controls discoverability of nodes and hence the established links are forced to meet the restrictions described in Section 4.1.

Master nodes are allowed only for inquiry mode. So that a master can establish link with either a slave node or a free node that is going to be slave. If the inquiry scan mode were allowed for a master node, it would have been slave of another master, resulting with a master/slave bridge, which is a violation of the restrictions on scatternet topology.

Bridge nodes are allowed for neither inquiry nor inquiry scan modes, hence making them unable to discover a new node and unable to be discoverable by another node. By preventing inquiry mode master/slave bridges are prevented and by preventing inquiry scan mode 2-master restriction for bridge nodes is obeyed.

Slave nodes are allowed only for inquiry scan mode. So that they can discover master nodes that they can potentially connect to as slave. Another possibility is that they connect to a free node that is going to be a master. If the inquiry mode were allowed for a slave node, it would have possibly been connected to another node being master of it, resulting with an undesired scatternet topology.

Free nodes, which are not yet connected to any other node, can be both in inquiry and inquiry scan modes. If a free node discovers another node when it is in inquiry mode, it becomes master, otherwise slave of the other node.

Unlike the *Maintenance* procedure, which is run only on master nodes, the *Link Establishment* procedure runs on any node. Nodes, except masters that already have seven slaves and bridges, are scheduled to enter one or both of the inquiry/inquiry scan modes, which is determined by the rules listed above, at certain intervals. These intervals are determined by the master node for itself and for its slaves. Once a new node is discovered in accordance with the restrictions on the topology of scatternet, the *Link Establishment* procedure further decides to establish the link or not to. Link establishment can be restricted in order to limit the average node degrees. One such restriction would be to prevent establishing links between a slave and the master of one-hop distance piconet. This information can be passed to slave nodes each time they are instructed to enter inquiry scan mode, by the master, which already have one-hop distance master information in its traffic table. Other controls can be imposed whether to continue with page/page scan mode after the discovery, using probabilistic approaches or acquiring 2-, 3-hop information by extra messaging if desired.

The *Link Establishment* procedure enables dynamic handling of new link establishments. So that the scatternet topologies that can further be modified by the *Maintenance* procedure for bandwidth-efficiency is constructed on the fly. During the *Maintenance* procedure the set of nodes constituting the piconet should not be changed, hence coordination between the *Link Establishment* and the *Maintenance* procedures is required. This can be achieved if the masters do not schedule the slave nodes for device discovery before beginning the *Maintenance* procedure (i.e. Algorithm 3).

Chapter 5

Evaluation Criteria and Simulation Results

In this chapter, first, the criteria used and next, simulation environment prepared for evaluating the performance of the proposed algorithm are presented. In the last section of the chapter, simulation results are discussed.

5.1 Evaluation Criteria

In order to evaluate the performance of the algorithm Weighted Average Shortest Path (WASP) approach proposed in [24] is used. WASP aims to reflect the load on the network for a given network topology and traffic demand distribution among nodes. Before explaining WASP in detail, concepts of end-to-end *traffic demand* and *traffic load* on a network should be made clear. End-to-end *traffic demand* is the rate of communication between two nodes. This communication imposes *traffic load* on the network and it is computed by multiplying the amount of *traffic demand* by the number of hops between demanding nodes. WASP of a traffic flow between two nodes, which is due to traffic demand between these nodes, is computed by dividing traffic load, which is created by related traffic demand, by total traffic demand of all nodes in the network. More formally, if

d_x is the traffic demand of node pair x, L_x is the number of hops between these two nodes and n is the number of all demanding node pairs, WASP of flow due to this demand, WASP_x, is;

$$\text{WASP}_x = \frac{d_x \times L_x}{\sum_{i=1}^{n} d_i} \tag{5.1}$$

WASP of the network is achieved by adding WASPs of all flows, which are due to traffic demands among all nodes. Hence WASP of a network is;

$$\text{WASP} = \frac{\sum\limits_{i=1}^{n} d_i \times L_i}{\sum\limits_{i=1}^{n} d_i} \tag{5.2}$$

Average Shortest Path (ASP), which is proposed in [25] as a performance metric, is computed in a similar manner: Number of hops between every demanding node pair is added and the result is divided by number of demanding pairs. Although in reality WASP is an extended version of ASP, ASP can be thought as a special case of WASP where every demand, d_x, is equal to 1. ASP simply shows average number of hops between any communicating node pair and is more intuitive than WASP, but it does not reflect effect of traffic demand of nodes. For example, although a node pair with $2d$ amount of traffic demand and distance of 2 hops has more load on the network than a node pair with d amount of traffic demand and distance of 3 hops, this fact is not reflected in ASP which makes WASP a better choice, since it also takes the traffic demands into account.

Another issue about evaluation is how to interpret the results and what to expect beforehand. Obviously WASP can have 1 or larger values, and it should be noted that, as the WASP value gets closer to 1, the traffic load on the network decreases, meaning that pairs with higher traffic demands are closer. If there is communication between any two nodes and the traffic demands of these pairs are all equal regardless of the network topology WASP gives the same result with ASP, hence the result is the average number of hops. For a given network size, as the variation of traffic demands increases, effect of changing the topology of the network on the results of WASP should also increase. Similarly for a given traffic demand variation, other than all-to-all and even (i.e. any node in the network has the same probability of communicating with any other node at certain rate), as the number of nodes increases, effect of topology change should also increase.

5.2 Simulation Environment

In order to test the performance of the proposed algorithm a simulation environment is prepared, which uses WASP as performance metric. Simulation is implemented in C++ programming language with an object oriented approach. The basic object relations is depicted in Figure 5.1.

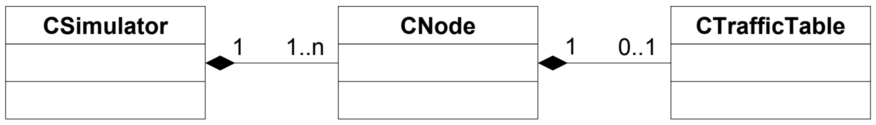


Figure 5.1: Basic object relations for simulation implementation

CNode represents a node in the network. Although all nodes are capable of executing the proposed algorithm and modifying the topology according to the result of it, only master nodes do so. Master nodes also have traffic tables, which they use as the input for the operations of the algorithm. CTrafficTable stores the traffic table entries of a piconet. It is also capable of computing the best configuration (i.e. configuration with minimum cost) for each operation of the algorithm. CNode uses this configuration information to modify the topology.

CSimulator is responsible for creation of nodes (i.e. CNode objects) and determining initial relations among them, that is the initial scatternet topology with the roles assigned to the nodes. The traffic demands among the nodes are also determined by this class. The initial scatternet topology can be generated either automatically for a given node count or manually. Similarly, the traffic demands can be generated randomly for given a traffic demand characteristic or manually. Once the initial scatternet topology along with the traffic demands is created, the master nodes are assigned their traffic tables which are extracted from the global traffic demand information.

Algorithm 4 General flow of the simulation

```
1:  for $nodeCount = size_{min}$ to $size_{max}$ with steps $size_{step}$ do
2:    Initial-$WASP_{nodeCount} \leftarrow 0$
3:    Final-$WASP_{nodeCount} \leftarrow 0$
4:    for $experimentCount = 1$ to 100 do
5:      Generate a toplogy of size $nodeCount$
6:      Generate traffic demands table according to a traffic characteristic
7:      $WASP_{current} \leftarrow$ WASP of current topology
8:      Initial-$WASP_{nodeCount} \leftarrow$ Initial-$WASP_{nodeCount} + WASP_{current}$
9:      repeat
10:       Assign traffic tables of master nodes
11:       $WASP_{previous} \leftarrow WASP_{current}$
12:       Choose a random master node $M$
13:       Let $M$ to execute the algorithm and modify the topology
14:       $WASP_{current} \leftarrow$ WASP of current topology
15:     until $WASP_{current}$ differs $WASP_{previous}$ less than 1%
16:     Final-$WASP_{nodeCount} \leftarrow$ Final-$WASP_{nodeCount} + WASP_{current}$
17:   end for
18:   Output (Initial-$WASP_{nodeCount}/100$) and (Final-$WASP_{nodeCount}/100$)
19: end for
```

The simulation environment is designed to examine the effect of the *Maintenance* procedure, which is the integral part of the algorithm in order to construct bandwidth-efficient topologies. Hence the simulation starts with an initial scatternet topology and runs the *Maintenance* procedure on the same set of nodes until a stabilized topology is reached. The traffic demands are also kept unchanged during this period. As mentioned before the evaluation criteria is WASP. In simulation environment two parameters are changed to see their effects on WASP:

- Number of nodes in the scatternet
- Traffic demand characteristics

Algorithm 4 shows the general flow of the simulation. The simulation runs for different scatternet sizes and for different traffic characteristics. Scatternet sizes are chosen to be between $size_{min}$ and $size_{max}$ with a step size of $size_{step}$.

For each scatternet size, 100 different initial scatternet topologies are generated and the proposed algorithm is run on arbitrary master nodes until no significant improvement on WASP is obtained. Numerically speaking, if the improvement is less than 1% compared to the previous topology, the scatternet is assumed to be stabilized. For each scatternet size, the WASP values for each of 100 initial and final (i.e. stabilized) topologies are summed up and their average is calculated. Additionally, for each scatternet size, average number of the *Maintenance* procedure executions, until the final topology is reached, is calculated, which is not shown in Algorithm 4.

5.3 Simulation Results

Similation environment described in previous section is run for different scatternet sizes and different traffic characteristics. This minimum and maximum scatternet sizes (i.e. $size_{min}$ and $size_{max}$) are decided to be 10 and 100 respectively, whereas the step size (i.e. $size_{step}$) is 10. On the other hand, three different traffic characteristics, TC-1, TC-2 and TC-3, are used in simulations. In first type of traffic characteristic, TC-1, given a node pair out of all possible node pairs, they

- Do not communicate with a probability of 0.3
- Communicate at 5 Kbps with a probability of 0.2
- Communicate at 10 Kbps with a probability of 0.2
- Communicate at 15 Kbps with a probability of 0.2
- Communicate at 20 Kbps with a probability of 0.1

In second type of traffic characteristic, TC-2, given a node pair out of all node pairs, they

- Do not communicate with a probability of 0.4

- Communicate at 5 Kbps with a probability of 0.3
- Communicate at 30 Kbps with a probability of 0.3

In third type of traffic characteristic, TC-3, nodes are grouped into three as Group$_A$, Group$_B$ and Group$_C$, which constitute 30%, 30% and 40% of scatternet respectively. Given a node pair

- Both from Group$_A$, they communicate at 20 Kbps with a probability of 0.7
- Both from Group$_B$, they communicate at 15 Kbps with a probability of 0.8
- Both from Group$_C$, they communicate at 15 Kbps with a probability of 0.6
- One from Group$_A$ and one from Group$_B$, they communicate at 4 Kbps with a probability of 0.2
- One from Group$A$ and one from Group$_C$, they communicate at 2 Kbps with a probability of 0.3
- One from Group$_B$ and one from Group$_C$, they communicate at 3 Kbps with a probability of 0.3

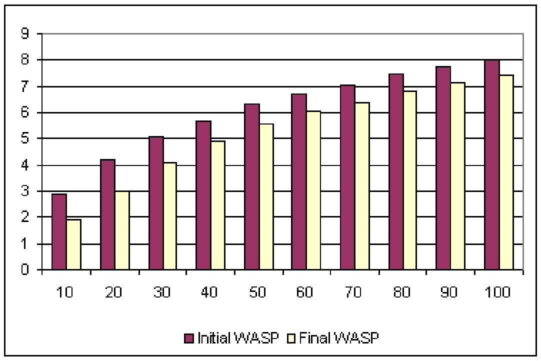
Note that, in TC-1 and TC-2 although there are different communication rates they are evenly distributes among node pairs. Different from TC-1 and TC-2, in TC-3, there are groups, which have higher communication rates within the group and low rates with other groups.

The initial and final WASP values for these three traffic characteristics on varying scatternet sizes are depicted in Figure 5.2 (a)-(c). As it is clearly seen, whatever the traffic characteristics is, the initial WASP values are very similar. However, as the scatternet-wide communication relations among nodes get weaker, the final WASP values decrease.

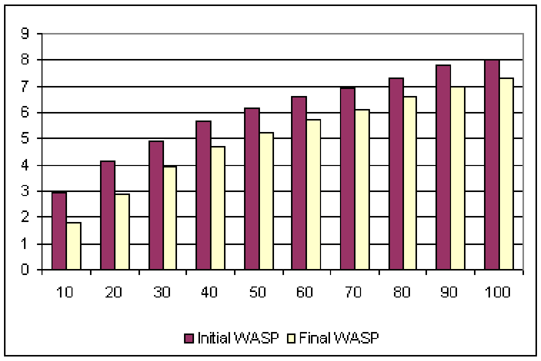
In Figure 5.3, improvements in WASP for TC-1, TC-2 and TC-3 are shown for varying scatternet sizes. For TC-3 the improvements are higher than TC-1 and TC-2, and improvements up to 46% can be reached, that is because the heavily communicating nodes can be grouped in the scatternet for TC-3. On the other hand, in TC-1 and TC-2 as a node is moved towards another node it highly communicates, it gets further from the other nodes it communicates at the same rate, since every node has a similar probability of communication with another node at certain rate. TC-2 has better improvement than TC-1, since the ratio of non-communicating nodes is higher in TC-2 and it has a more diverse traffic distribution than TC-1. Another observation from Figure 5.3 is that as the scatternets get larger the improvements decrease. This is due to the increasing communication rate per piconet. As the number of communicating nodes increases total traffic demand and consequently the load on the scatternet increases, meaning that a piconet should relay more traffic. According to the proposed algorithm the piconets are divided as the communication rate that they are expected to handle is above a threshold. Due to the situation described above, as the scatternet gets larger piconets tend to divide, which increases the path length and consequently WASP. Although improvement in WASP decreases, since the average available bandwidth per piconet on the way from source to destination increases, the data packets are not dropped due to high congestion, which, in fact, means improvement in general performance.

As explained before, during the simulation the proposed algorithm is run on randomly chosen master nodes until the scatternet is stabilized. Figure 5.4 shows average number of executions of the algorithm for different scatternet sizes and traffic characteristics. Although the improvements are better in TC-3 as discussed, it has higher number of algorithm executions until the network is stabilized. That is because traffic characteristics similar to TC-1 and TC-2 are less appropriate to improvements due to their even scatternet-wide traffic distribution. It should be noted that the actual number of algorithm executions would be much less than the simulation results since in simulation, the algorithm is executed at randomly chosen piconets, even if the algorithm execution will result with no effect, whereas in reality the master nodes will decide to execute the algorithm when necessary by observing significant changes in piconet-wide traffic. The results reflecting the number of executions of the algorithm are given to show the general behavior.

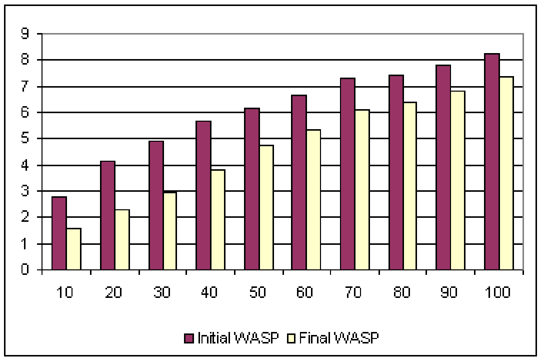
The simulation environment is also used to investigate the differences between the initial and final scatternet topologies. Scatternet, which has a chain topology, shown in Figure 5.5 is converted into the topology shown in Figure 5.6 (a) in four algorithm executions where each node pair has a communication rate of 3 Kbps. The initial WASP is 3.3 and the final WASP is 2. After the algorithm is applied, piconets are utilized as much as possible as it can be seen from the figure. Starting from the same chain topology but this time with traffic demands among the nodes increased to 8 Kbps per node pair, the final topology depicted in Figure 5.6 (b) is reached. This is due to the threshold value for maximum communication rate within a piconet. The final WASP value this time is 2.18.



(a)



(b)



(c)

Figure 5.2: Initial and final WASP values for (a) TC-1, (b) TC-2 and (c) TC-3

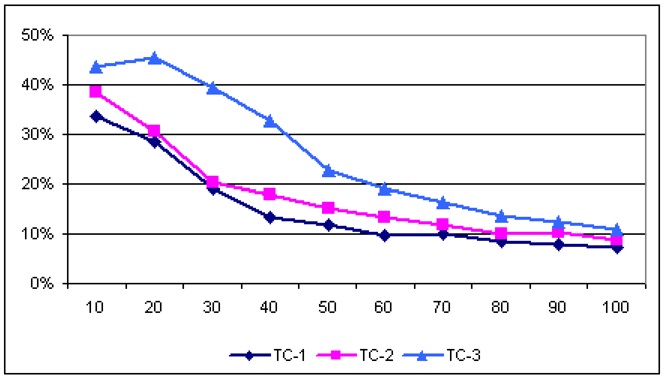


Figure 5.3: Improvement in WASP for different traffic characteristics.

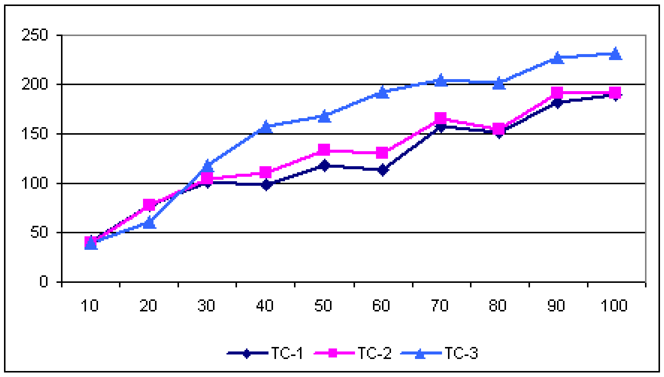


Figure 5.4: Number of algorithm executions for different traffic characteristics.

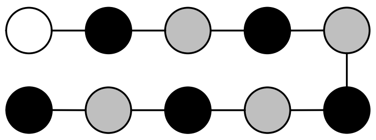
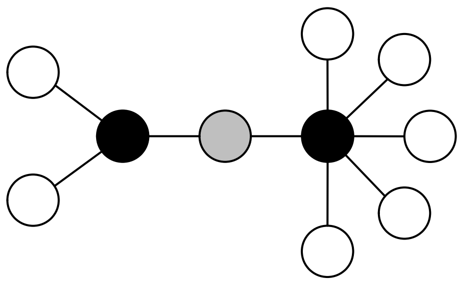
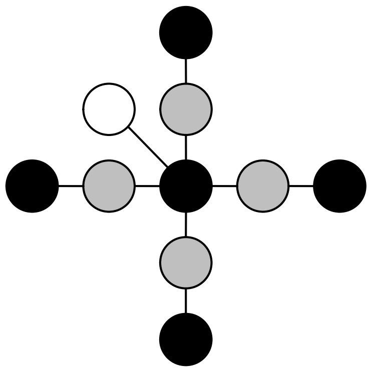


Figure 5.5: Initial scatternet topology



(a)



(b)

Figure 5.6: Final scatternet topologies at communication rates of (a) 3 Kbps and (b) 8 Kbps

Chapter 6

Conclusion and Future Work

6.1 Conclusion

A novel scatternet construction and maintenance algorithm, which aims to minimize the bandwidth usage in the resulting scatternet topology, is proposed in this study. The algorithm proposed in the thesis differs from many of the previously proposed scatternet construction algorithms at its basic motivation. It is different from algorithms having similar motivation too with its distributed approach for dynamic environments.

The algorithm is composed of two procedures, which run concurrently. As one of them handles establishment of new links, the other modifies the current topology in order to achieve a better state in terms of bandwidth efficiency. The proposed algorithm works in a distributed fashion and it requires minimal messaging overhead. It is also adaptive, meaning that it preserves bandwidth efficiency as traffic demands change by changing the scatternet topology accordingly and can cope with arriving and leaving nodes by extending or narrowing the traffic tables appropriately. As the bandwidth usage is reduced in the scatternet, available bandwidth for new communication demand increases, for sure. Furthermore, since the average number of hops between communicating nodes and the traffic load on the paths are reduced, the average latency is also reduced consequently.

Another benefit, which has importance especially for mobile nodes, is that the average power consumption per node decreases, since the amount of data that is not directly related with the node but still needs to be forwarded (i.e. relayed traffic) reduces. Success of the proposed algorithm is also evaluated and the results are presented in the thesis. Simulation results show that the approach used in the algorithm is promising.

6.2 Future Work

Although the proposed scatternet construction and maintenance algorithm is designed to meet the common needs of solutions in wireless ad-hoc networks such as distributed approach with as minimum messaging overhead as possible and adaptivity to changing conditions, there are still points that may contradict with real life situations. One of such situations is related with one of the assumptions of the algorithm, which requires all the nodes should be in the communication range of each other. Although this assumption is fair in many cases like students in a classroom, audiences in a conference room etc. there will certainly be times this is not the case. Hence, the proposed algorithm needs to be modified in order to handle such cases. As far as the operations are concerned, during calculation of cost functions for each possible configuration of roles, infeasible configurations must be eliminated. Although such a modification for each operation is rather easy, determining visibility information for each node is a costly process requiring constant check of neighbors for each node. Furthermore, extra messaging is required to gather this information at master nodes, which are the decision points in the algorithm. Performance of the modified algorithm should also be investigated to see whether it is worthy to apply the algorithm in such a scenario. As another study on the algorithm, rather than using exact cost calculations in the operations, some clever heuristics can be developed to perform the operations in order to reduce the processing power requirements of the algorithm. Although the cost calculations are not a big deal for most of the current mobile devices, offering choice between heuristics and exact calculations will make the algorithm deployable even to devices with very little processing capability.

Bibliography

[1] AU-System. *Bluetooth White Paper 1.1*, January 2000.

[2] S. Baatz, C. Bieschke, M. Frank, C. Kuhl, P. Martini, and C. Scholz. Building efficient Bluetooth scatternet topologies from 1-factors. In *Proceedings of the IASTED International Conference on Wireless and Optical Communications, WOC 2002*, Alberta, Canada, July 2002.

[3] C. Bisdikian. An overview of the Bluetooth wireless technology. Technical Report RC 22109, Thomas J. Watson Research Center, IBM Research Division, June 2001.

[4] Bluetooth Special Interest Group, http://www.bluetooth.com. *Bluetooth.*

[5] Forum Nokia. *Bluetooth Technology Overview*, April 2000.

[6] J. Ghosh, V. Kumar, X. Wang, and C. Qiao. Btspin - Single phase distributed Bluetooth scatternet formation. Technical Report 2004-06, Department of Computer Science and Engineering, University at Buffalo, The State University of New York, December 2003.

[7] IEEE 802.11 Working Group, http://grouper.ieee.org/groups/802/11/. *IEEE 802.11 Specifications.*

[8] Infrared Data Association, http://www.irda.org. *IrDA Specifications.*

[9] M. Kalia, S.Garg, and R. Shorey. Scatternet structure and inter-piconet communication in the Bluetooth system. In *IEEE National Conference on Communications*, New Delhi, India, 2000.

[10] R. Kapoor, M. Sanadidi, and M. Gerla. An analysis of Bluetooth scatternet topologies. In *ICC 2003*, volume 1, pages 266–270, Anchorage, AK, USA, May 2003.

[11] J. Kardach. Bluetooth architecture overview. *Intel Technology Journal*, 2000.

[12] Y. Kawamoto, V. Wong, and V. Leung. A two-phase scatternet formation protocol for Bluetooth wireless personal area networks. In *Wireless Communications and Networking, 2003*, volume 3, pages 1453–1458, March 2003.

[13] C. Law, A. K. Mehta, and K. Y. Siu. Performance of a new Bluetooth scatternet formation protocol. In *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing*, pages 183–192, Long Beach, CA, USA, October 2001.

[14] C. Law and K. Y. Siu. A Bluetooth scatternet formation algorithm. In *Proceedings of IEEE Globecom 2001*, pages 2864–2869, San Antonio, Texas, USA, November 2001.

[15] M. A. Marsan, C. F. Chiasserini, A. Nucci, G. Carello, and L. D. Giovanni. Optimizing the topology of Bluetooth wireless personal area networks. In *IEEE INFOCOM 2002*, pages 572–579, NY, USA, June 2002.

[16] G. Miklos, A. Racz, Z. Turanyi, A. Valko, and P. Johansson. Performance aspects of Bluetooth scatternet formation. In *Proceedings of the 1st ACM International Symposium on Mobile Ad Hoc Networking & Computing*, pages 147–148, Boston, Massachusetts, USA, November 2000.

[17] D. Miorandi, A. Trainito, and A. Zanella. On efficient topologies for Bluetooth scatternets. *Lecture Notes in Computer Science*, 2775/2003:726–740, 2003.

[18] D. Miorandi and A. Zanella. On the optimal topology of Bluetooth piconets: Roles swapping algorithms. In *Proceedings of Mediterranean Conference on Ad Hoc Networks, Med-Hoc-Net*, Sardinia, Italy, September 2002.

[19] C. Petrioli, S. Basagni, and I. Chlamtac. Configuring BlueStars: Multihop scatternet formation for Bluetooth networks. *IEEE Transactions on Computers, Special Issue on Wireless Internet*, 52(6):779–790, 2003.

[20] T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire. Distributed topology construction of bluetooth personal area networks. In *Proceedings of the INFOCOM 2001*, volume 3, pages 1577–1586, Anchorage, AK, USA, April 2001. IEEE.

[21] H. Sreenivas and H. Ali. An evolutionary Bluetooth scatternet formation protocol. In *Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04)*, pages 306–313, Big Island, Hawaii, USA, January 2004.

[22] G. Tan, A. Miu, J. Guttag, H. Balakrishnan, T. Berners-Lee, L. Masinter, and M. McCahill. Forming scatternets from Bluetooth personal area networks. Technical Report MIT-LCS-TR-826, MIT Laboratory for Computer Science, October 2001.

[23] G. Tan, A. Miu, J. Guttah, and H. Balakrishnan. An efficient scatternet formation algorithm for dynamic environments. In *Proceedings of the IASTED Communications and Computer Networks (CCN)*, Cambridge, MA, November 2002.

[24] T. Topal. Constructing efficient bluetooth scatternets. Master's thesis, Department of Computer Engineering, Bilkent University, 2004.

[25] Z. Wang, R. J. Thomas, and Z. Haas. Bluenet - A new scatternet formation scheme. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, volume 2, January 2002.

[26] J. Yun, J. Kim, Y.-S. Kim, and J. Ma. A three-phase ad hoc network formation protocol for Bluetooth systems. In *Proceedings of the 5th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, volume 1, pages 213–217, Honolulu, USA, October 2002.

[27] G. Zaruba, S. Basagni, and I. Chlamtac. BlueTrees - Scatternet formation to enable Bluetooth-based personal area networks. In *Proceedings of the IEEE International Conference on Communications, ICC 2001*, Helsinki, Finland, June 2001.

[28] S. Zurbes. Considerations on link and system throughput of Bluetooth networks. In *Proceedings of the 11th International Symposium on Personal, Indoor and Mobile Radio Communications*, volume 2, pages 1315–1319, London, UK, September 2000. IEEE.

Appendix A

Table of Acronyms

AC	Access Code
ACL	Asynchronous Connectionless link
AM_ADDR	Active Member Address
ARQ	Automatic Repeat Request
ASP	Average Shortest Path
AT	Attention
BD_ADDR	Bluetooth Device Address
CRC	Cyclic Redundancy Check
FEC	Forward Error Correction
FHS	Frequency Hop Sequence packet
FHSS	Frequency Hopping Spread Spectrum
GFSK	Gaussian Frequency Shift-Keying
GHz	Gigahertz
HCI	Host Controller Interface
Kbps	Kilobits per second
IP	Internet Protocol
ISM	Industrial, Scientific and Medical band
L2CAP	Logical Link Control and Adaptation Protocol
LAN	Local Area Network
LMP	Link Manager Protocol

Mbps	Megabits per second
MHz	Megahertz
MTU	Maximum Transferable Unit
PM_ADDR	Parked Member Address
PPP	Point-to-Point Protocol
QoS	Quality of Service
RF	Radio Frequency
SCO	Synchronous Connection-Oriented link
SDP	Service Discovery Protocol
SIG	Special Interest Group
TCP	Transmission Control Protocol
TCS	Telephony Control Signalling
TDD	Time Division Duplex
TDMA	Time Division Multiple Access
UDP	User Datagram Protocol
WAP	Wireless Application Protocol
WASP	Weighted Average Shortest Path