Arguments for Cross-Layer Optimizations in Bluetooth Scatternets

Bhaskaran Raman*
EECS Department, U.C.Berkeley
bhaskar@cs.berkeley.edu

Pravin Bhagwat*
AT&T Labs Research
pravinb@research.att.com

Srinivasan Seshan*
School of Computer Science, CMU
srini.seshan@cs.cmu.edu

Abstract

Bluetooth, a recent innovation in short-range radio technology, has gone through the first stage of standardization, and commercial products based on v1.0 specifications will be appearing soon. While much work has gone into developing the radio technology and hardware for this system, little effort has been focused on additional infrastructure that is necessary for applications in this environment. In this paper, we examine the issue of supporting ubiquitous computing applications [21, 14] in a Bluetooth network.

The Bluetooth standard defines a multi-hop routing structure, called a scatternet, to address the limitations caused by short-range and small fanout of the underlying link technology. We identify several characteristics, the combination of which makes scatternets different from previously considered networks. Importantly, Bluetooth links are connection-oriented with low-power link modes. We show that the unique aspects of the technology require a redesign of the protocol structure for link formation, IP routing, and service discovery. When existing approaches to these protocols are applied to scatternets, the multiple protocol layers would operate without knowledge of each other, resulting in inefficient use of power in many cases. We suggest an alternative approach where there is a single protocol layer providing a level of indirection within the scope of a scatternet. That is, we argue for extensive cross-layer optimizations. Specifically, this allows (a) links to be kept active only when absolutely required, and (b) scatternet-wide floods to be minimized by caching service discovery results at all intermediate nodes. Our experiments with an implementation of an emulated Bluetooth scatternet show that this could be more efficient than the traditional approach of layered protocol design.

*This work was done when the authors were at IBM T. J. Watson Research Center

1. Introduction

The vision put forth by a variety of ubiquitous computing projects [21] has created a strong demand for a low cost, wireless communication technology. This demand has led to the development of the Bluetooth radio system [1]. Due to the recent release of the standard (July 1999) and the strong commercial support, Bluetooth radio links are expected to be available in a wide variety of products in the near future. In this paper, we examine the demands placed on Bluetooth networking technology by ubiquitous computing applications and explore some of the critical design tradeoffs that must be made in such a system.

The basic structure for communication in a Bluetooth network is a *piconet*, an interconnect between a *master* device and one to seven *slave* devices. The limited range of a single link and fanout limit of a piconet prevent all devices in a region from communicating directly with each other. To address this issue, the Bluetooth system allows multiple piconets to be composed into a multihop wireless *scatternet* (see Figure 1). However, the current specifications [1] do not describe the algorithms or mechanisms to create a scatternet due to a variety of unsolved issues.

Bluetooth networks are significantly different from other networks in which ubiquitous computing has been explored. These differences lie in five areas:

1. *Spontaneous network*. Bluetooth networks are formed "spontaneously", without any pre-planning. As a result, applications and hosts must be able to auto-configure and adapt to changing membership [2].

2. *Isolation*. A Bluetooth network must be able to operate in isolation and cannot rely on infrastructure-based services such as DNS.

3. *Simple devices*. Bluetooth is targeted for use in low end, low cost devices and has been designed with stringent power considerations in mind.

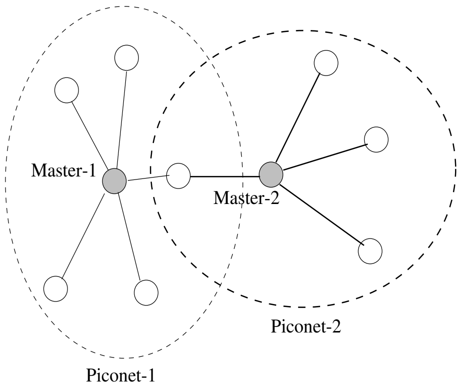

Figure 1. A scatternet with two piconets

4. *Small, multi-hop networks*. The primary application for Bluetooth networks is the local interaction between devices and users. Scatternets are not expected to have a diameter of greater than 5-10 hops.

5. *Connection-oriented, low-power link technology.* Each point-to-point link in Bluetooth has state. Such a design allows the definition of low power modes where nodes save power by sleeping most of the time. This is by far the most distinguishing feature of Bluetooth that differentiates it from other networks considered so far.

Although researchers have addressed a variety of issues in networks with some of the above characteristics [13, 10, 6, 3, 12, 18, 9], the approaches do not apply when *all* of the above characteristics are considered. The first three characteristics render any solutions in infrastructure based networks (with relatively high-end nodes) inapplicable. The fourth characteristic, the multi-hop nature, means that approaches that rely on traditional LAN emulation could be very inefficient. Finally, and most importantly, the last characteristic means that there is an explicit *link-formation* stage in scatternets.

We wish to stress that it is the combination of the above characteristics, and especially the 5th one, that makes scatternets different from other networks considered previously.

Our original goal was to build the specific application of *Lego-Computing*, which allows a user to combine a collection of wireless networked peripherals into a single "virtual computer". When such an application has to run on a scatternet, the network has to provide (at least) three mechanisms: *link-formation*, *IP-routing*, and *service-discovery*.

Applying existing solutions to these three requirements would result in a traditional **layered approach** with piecewise solutions. We suggest an alternative **integrated approach** that allows extensive cross-layer optimizations across the solutions to these three problems. Specifically, the integrated approach allows (a) links to be kept active only when absolutely required, and (b) scatternet-wide floods to be minimized by caching service discovery results at all intermediate nodes.

We have an implementation of the *Lego-Computing* application in an emulated scatternet. We instrumented this to quantify the inefficiencies in a layered approach. Our studies show that the integrated approach can be more efficient in terms of the number of messages and the power consumed in many usage scenarios.

The rest of the paper is organized as follows. The next section discusses the unique characteristics of scatternet environments, and the requirements for application support. Section 3 presents a brief overview of the current approaches to meeting these requirements in related environments. Section 4 presents our arguments against extending existing protocol layering to scatternets and Section 5 quantifies these arguments. Finally, we summarize our observations and discuss their implications in Section 6.

2. Requirements of Scatternet Applications

Bluetooth technology is expected to enable the networking of a wide variety of user devices and computer peripherals. In this section, we describe a motivating application, Lego-Computing, and use this description to arrive at the mechanisms required to support such applications.

2.1. Lego-Computing

Lego-Computing is an application where a "virtual computer" can be assembled on the fly using wireless computer peripherals. The original idea behind this comes from the "Virtual Network Computing" (VNC) project [7]. VNC allows a user to access a desktop session from a remote computer. With lego-computing, in addition to the session transfer, the user "constructs" the computer (for access to the remote session) on the fly using wireless I/O peripherals.

Imagine the scenario where a user walks into a room, uses her PDA to locate the necessary I/O peripherals, and interconnects them to form a "lego-computer". She then uses the lego-computer to access the session at her office computer (a session in the VNC sense means all desktop state: same windows, same window positions, etc.). The I/O peripherals could be wireless keyboard, mouse, or display. And the connectivity to the remote session could be provided by a network access point in the room.

2.2. Requirements

For enabling the Lego-Computing (and similar) applications, we need at least the following three crucial functional capabilities in a scatternet.

1. *Link Formation*. Unlike other wireless technologies like 802.11, physical proximity does not mean the existence of a link between two Bluetooth nodes. There is an explicit link-formation stage. The link has state – Bluetooth defines active, sniff, hold, and park modes for a link [4]. Link-formation refers to a transition from an inactive state (low power) to an active state (high power) in which packets can be exchanged. A protocol is required to *create* the scatternet topology on top of which packets can be routed.

2. *IP layer routing*. Once links are created, the various nodes must cooperate and forward packets. This requires the nodes to provide some IP-like routing mechanism within the scatternet.

3. *Service-discovery (SD)*. Finally, service discovery is needed for nodes to identify who they wish to communicate with. Since pre-configuration is not an option in scatternet environments, service discovery is even more crucial than in relatively static environments.

The link-formation mechanism has to include algorithms for deciding (a) *which* links to form, (b) *when* to form them, and (c) when a link should be *torn-down*. The IP layer in a scatternet may also include an automatic address allocation mechanism [20]. We have not included host-name mapping as a separate requirement since a flexible service discovery mechanism can achieve this functionality. And, by service discovery we only mean the protocol aspects and not the query representation and language aspects that are somewhat orthogonal (albeit important).

3. Current Approaches

Solutions to the requirements listed above have traditionally been implemented in a layered fashion. In this section, we review some of the existing solutions and especially look into recent work that has explored gains from more integrated approaches. We also point out where we leverage protocols and ideas from existing work.

The classical connection-oriented link-technology over which IP routing has been operated is ATM [15]. To hide link-formation from higher layers, LAN emulation is done on top of a set of ATM nodes – Virtual Circuits (VC) are made on-demand and an ATMARP server is used. This is possible since ATM networks are relatively static. Such an infrastructure based approach for link-formation does not apply to scatternets. Furthermore, ATM VCs come at little extra cost while battery power has to be spent for Bluetooth link maintenance.

Techniques for IP-style routing of messages in dynamically formed networks have been studied extensively [9, 16, 18, 17, 19]. We leverage the on-demand protocols from this work [18, 9] and study how these can be made more efficient using cross-layer optimizations. We also study the interaction of service discovery with these protocols.

Current approaches to service discovery use either a dedicated centralized server, an existing level of indirection such as DNS, and/or a broadcast/multicast search [3, 13, 10]. The approaches that depend on a centralized server cannot be used in isolated scatternets. The IETF standardized Service Location Protocol (SLP) [13] uses scoped multicast to find services in small networks. So does the Simple Service Discovery Protocol (SSDP) [12]. This *style* of service discovery is suitable for scatternets. However, the implicit assumption made by these protocols is the presence of the layers below: they assume IP broadcast/multicast as well as a pre-existing link layer. That is, they take a *layered* approach to solving the problem.

Recent work on "Intentional Naming System" (INS) [6] has proposed combining the service discovery layer with the routing layer (i.e., application level routers) to meet application requirements. We leverage this idea of generic routing as opposed to IP-address based routing. However, since this solution has been proposed for the Internet, it assumes infrastructure, and existing solutions like DNS for bootstrap. In addition, this study was done in an environment where explicit link formation is not needed.

Cross-layer optimizations have been considered in other scenarios such as for efficient TCP over wireless media [8]. Such cross-layer optimizations have so far not been considered for efficient application support in scatternet-like networks.

The current Bluetooth specifications describe a Service Discovery Protocol (SDP) [5] which is tightly integrated with link formation. However, this solution is only applicable to single-hop point-to-point links. A mechanism to propagate this service information across multiple hops has not been defined. The suggestions in this paper can provide a framework to extend SDP to multi-hop scatternets.

4. The Case for an Integrated Approach

In this section, we explore the advantages of an integrated approach over a layered approach. Our arguments are based on the following four key observations.

4.1. On-demand operation

For a single-hop, connection-less link layer like Ethernet or 802.11, there is no link-maintenance cost. Even with connection-oriented link layers like ATM, the cost of link-maintenance is negligible. Similarly, at the IP-layer, the additional cost in maintaining IP routes is insignificant compared to the benefits of doing this.

However, in a scatternet, links should not be kept active for long periods of time to conserve power. Likewise, keeping IP routes active can also be inefficient. This latter observation has also been made in previous work [18, 9].

The reason why on-demand operation is crucial in scatternets is that the usage model of scatternets is likely to follow a pattern of long periods of inactivity interspersed with brief periods of activity: imagine a set of devices in a room that are activated only when a user walks in and triggers an application. Such a pattern of activity can be expected to be typical of many home or office scenarios. It makes little sense to have the network actively exchange information during periods when there are no users of services.

This on-demand nature of operation, the fact that lower layers operate only on being triggered by a higher layer, gives the first hint at possible cross-layer optimizations.

4.2. Awareness of higher layer requirements

Fundamentally, applications look for services, not for IP-addresses. This is especially so in ad-hoc zero-configuration application scenarios [2]. The "demand" for on-demand operation comes in the form of a service request from the user. The way the lower layers function should be determined by what the application requires. This suggests that having some information exchange across the layers would be of use. Specifically, the link-layer can use information about the "demand" to decide *whether or not* to form a particular link, or to decide *which* of a set of links to form.

Consider the scenario depicted in Figure 2. Node C is a client looking for a service S; $N1$ and $N2$ are two other nodes. All nodes are in physical proximity of each other. The link formation layer has to decide which links to form.

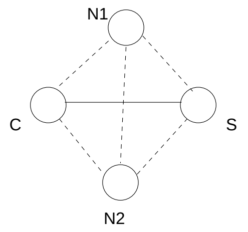

Figure 2. Use of SD info in link formation

Without any information on what the layer above requires, the algorithm to form links would be direction-less. It could result in an inefficient route between C and S. Worse, it could unnecessarily activate extra links (the dotted ones in the figure), effectively voiding all the effort that has gone into the careful definition of low power link modes.

More important than deciding when to form links, an integrated approach would allow to decide when *not* to keep a link active. Consider the scenario in Figure 3. The scatternet is inactive to begin with and there are no links. A node in the middle of the scatternet, C wants to find the service S. It has to flood/broadcast in the scatternet with a service request to locate S - which responds with a (unicast) reply. Links are formed on-demand to propagate the flood.

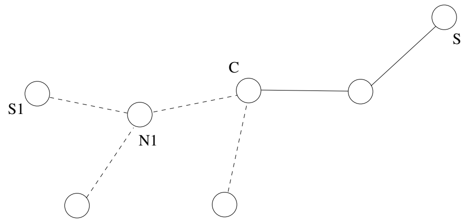

Figure 3. Minimizing link active mode time

The links to the left of C in the figure are all formed wastefully because no traffic will ever flow through these links. In a layered implementation, the best option is to timeout these links after a period of inactivity. However, when lower layers are aware of the semantics of higher layers, optimizations are possible. If a service *request* is being sent on a link, the link is brought back to inactive state immediately after transmission - since the probability that the reply to the flood is going to come back on that link is very low. However, if a service *reply* is being sent, the link is kept active even after transmission, anticipating further traffic. This way a flood will not keep the links active for long; only the reply will and only the links in the relevant portion of the scatternet become active.

Such optimizations are especially of importance in scenarios where some nodes are accessed more frequently than others. For instance, imagine a time-sync service that is activated by the user's watch for synchronization periodically, and a temperature sensor in the room that gets activated only when she wishes to read it using her PDA (much less frequently). When the frequent-use devices are activated, the others need not be disturbed unless absolutely essential (such as for connectivity). In Figure 3, suppose $S1$ is a service that is rarely accessed, there is no reason for $N1$ to keep any of its links active, except for short durations to forward floods. In an integrated approach, $N1$ would activate its link only on a service *reply* - which of course it would see only when C accesses $S1$.

These optimizations can possibly be done by an appropriate interface between the link and IP layers, including directives from the IP layer to change the link state. But there could be many possible power levels corresponding to link states, and the characteristics of each would be specific to the link technology. This would make the interface definition specific to the link technology – tending towards an integrated realization of the layers.

4.3. Caching of service descriptions

Consider the two functionalities of IP-routing and service-discovery. In a layered approach, the service discovery layer would operate without knowledge of the IP layer. For instance, SLP [13] or SSDP [12] propose using an IP broadcast/multicast mechanism to find services in a configuration-less environment. One could imagine layering the service discovery on top of the flooding functionality provided by an on-demand IP layer protocol like AODV [18] or DSR [9].

IP broadcast or multicast would come for free in a single hop network like an Ethernet or 802.11 since they are connection-less and there is no link maintenance cost. This is not the case with an unconfigured multi-hop scatternet operating on-demand. Here broadcast or multicast would amount to flooding.

Consider the possibility shown in Figure 4. A service discovery (SD) flood is initiated by client $C1$ looking for service S. The SD query is sent in terms of an IP broadcast. Node S replies to $C1$ over unicast and $C1$ now has the service information. Then $C2$ initiates a search for the same service S. In a layered approach, the second search would also involve an IP broadcast. However, in an integrated protocol, this seconds search would end at $N1$, and would not result in a broadcast. This is because the SD layer (or, the now integrated layer) would have the necessary information cached from the previous reply.

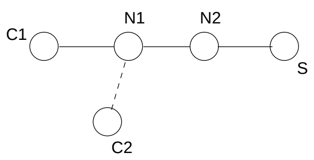

Figure 4. Integrated approach to minimize floods

Minimizing broadcasts saves power in two ways: (1) fewer messages are sent, and (2) fewer links are activated – this means that there are benefits to combining SD and IP layers even if the link-formation layer is kept separate.

Caching of service descriptions at relay nodes could also reduce response time of service queries. This is easily seen from Figure 4. Response time reduction in this manner is especially significant in Bluetooth since link establishment and reactivation delays could be high.

4.4. Same scope of operation

Yet another good reason why protocol functionality is strictly layered in traditional networks is that different layers have different scopes of operation. The link layer has the scope of a subnet, the IP layer has Internet-wide scope, and the SD layer could have administrative scope or wider.

Even within a layer, solutions that operate at different scopes tend to use different techniques. For example, wide area IP routing uses BGP while local area routing uses OSPF. The static nature of the network allows very specific optimizations to be used to solve problems at different scopes. IP routing uses the fact that nodes do not move and addresses are allocated hierarchically to aggregate and disseminate routing information. This optimization is critical to allowing routing to scale to the entire Internet. However, the optimizations unique to a layer prevent similar techniques from being used at other layers. For example, service discovery does not have structured names like IP addresses and cannot use the same optimization. Thus it does not make sense to integrate these layers.

However, in a scatternet, service names or IP addresses have no structure and are limited to the same scope. To find a service or to route to an IP address, both require a level of indirection in terms of a flood that is scatternet-wide. Both could use similar caching mechanisms for optimizations. Because of these strong similarities, an obvious optimization is to merge these mechanisms in a single layer rather than replicate the flooding functionality.

4.5. Shortcomings of Cross-Layer Optimizations

Although we have shown that a variety of cross-layer optimizations are possible in Bluetooth scatternets, we must also acknowledge that there are many advantages to a layered approach. Layered implementations tend to be modular – it is easy to upgrade or modify portions of an implementation. For example, it possible to change the format of service descriptions without impacting the behavior of the routing protocols. Since the different modules and layers are not tightly integrated, their correctness can be verified independently. Finally, modular programs also lend themselves to code reuse. Lower level building blocks can be used by a variety of upper layer protocols. These benefits outweigh the limited optimizations that are possible in traditional systems. However, as we have argued, the benefits of cross-layer optimizations in scatternets are significant enough to justify their use.

5. Quantitative Comparisons

In this section, we present a quantitative comparison of the different layering approaches. We consider specific usage scenarios and run simulations to show that there are a wide range of cases where a layered approach performs poorly.

5.1. The cases under comparison

Figure 5 shows the four possible ways of layering SD, IP, and link-formation. Integration of link-formation with a higher layer means that the link-formation stage uses information or directives from the higher layer to make intelligent decisions. Integration of the IP-routing layer with the SD layer means that both use the same flooding level of indirection (the functionality is not replicated).

In Figure 5, case(A) represents a fully integrated approach where all cross-layer optimizations are possible and case(D) represents a fully layered approach. In our comparisons, we also include an intermediate case(B) where the SD and IP layers are integrated, but the link-formation layer is separate. Case(C) represents another possibility where the SD layer is separate, but the IP and link layer are integrated. We do not consider this case for comparisons since we only consider scenarios where activity is triggered by service-discovery. In such cases, information from the SD layer is required for intelligent link-formation – there are no benefits to integrating the link-formation layer with the IP layer alone.

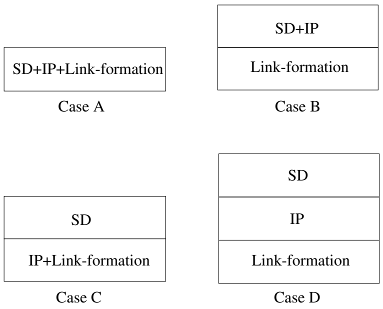

Figure 5. Different layering possibilities

In the fully integrated approach, there is a single broadcast-based request-reply algorithm for the SD and IP layers that is used for routing. We use the AODV [18] algorithm for this. Nodes learn the next-hop for routing to destination (IP-address or a service) with the mechanism of a request-flood and a unicast-reply. In the case when the SD layer is separate from the IP layer, we assume that the SD layer uses an IP broadcast primitive, like SSDP [12] does. (As noted earlier, broadcast, multicast, and flooding are the same in a spontaneous multihop environment). In this layered case, the IP routing layer uses the same protocol (AODV) as in the integrated case.

We have chosen an on-demand approach in accordance with the expected scenario of operation – home/office networks or similar environments – where we expect (a) long periods of inactivity interspersed with brief periods of activity, and (b) only a subset of the devices being active during the periods of activity.

With respect to link-formation, in the integrated approach, links are formed on-demand triggered by an SD query and only along the path of the query reply (see Section 4.2). The link layer in the integrated approach immediately returns to low power (or inactive) mode after sending, receiving, or forwarding an SD query flood. In the layered case, all links are brought to active state on a flood. In either case, a link has a timeout period. It is returned to low power mode (or dropped) if no packets are seen in this time. We fix this link timeout period for all our experimental runs (although it is possible to use an adaptive algorithm).

5.2. The Simulation Setup

The implementation of the Lego-Computing application was done on an "emulated" scatternet - emulated since we did not access to Bluetooth hardware. To study the behavior of the different layering approaches, instead of going for a full simulation, we instrumented the implementation. Bluetooth nodes are (unix) processes; a TCP connection between two nodes indicates physical proximity. A link between two nodes can be in two states: active or inactive. The Bluetooth node processes implement the integrated protocol layer with SD, IP-routing, and on-demand link formation. For the comparison studies, we also added support for separate IP routing and link-formation layers. These were implemented as described in Section 5.1.

Nodes know only about their neighbors to begin with. Routes to other nodes are found on-demand. Packets can originate from any node and travel along the network routed by intermediate nodes.

We consider application scenarios where a collection of nodes look for services hosted at other nodes in the network. We study activity with respect to SD queries and replies. We do not have topology changes during simulation runs – since Bluetooth has been designed for day-to-day devices at office or home, high node mobility during a session of activity is unlikely to be the common case, although we cannot rule it out. We timeout all information at the SD & IP layers periodically. The behavior during a timeout will be an approximation of the behavior during topology changes.

For the experiments below, we have chosen a value of $10sec$ for the SD or IP layer information timeout periods. And a lower value of $1sec$ for the link timeout period. The higher the link timeout period, the more the power wasted due to unnecessary link formation. We have a series of service queries for random services originating from random nodes at an aggregate rate of approximately 3/sec (approximate since it is an emulation and not an event driven simulation). We also tried various other values for all of these parameters; in all cases, the results exhibit the same trend.

Parameters: The parameters that we vary in the simulations are: (a) n, the number of nodes in the scatternet, (b)

M, the number of links that can be formed, (c) S, the number of services accessed, (d) nQ, the number of SD queries. The parameter M is a measure of the density of the topology. Note that all M links need not be active. The parameter S is not the total number of services that are possible in a scatternet, but the number of services that are accessed during a particular period of activity.

Metrics: We measure two quantities: (a) the time for which all the links of the scatternet taken together are active, and (b) the total number of messages in the network. Both are measures of the power consumed by the collection of nodes in the network. We keep track of these metrics as the simulation proceeds.

5.3. Results

To start with, we set *M=2n*, have only one service *S = 1* in the network, and issue a set of queries for the service from random nodes in the network. For this simulation, Figure 6 shows the first metric, the number of messages, for cases (B) and (D) of Figure 5. For both the cases, the metric is shown as a ratio over the number of messages that was seen with a fully integrated approach (case (A) of Figure 5). The x-axis is the number of queries issued for the service up to that point in the simulation. The graph shows the measurements for different values of n (5, 10, and 20). The measurements are the averages over four runs - two sets of random queries with two random topologies. The error-bars give the standard-error. We did not have more runs since the error-bars are small in most cases. Besides, we're not interested in any exact calculation but only in the rough measure of performance overhead in a layered approach.

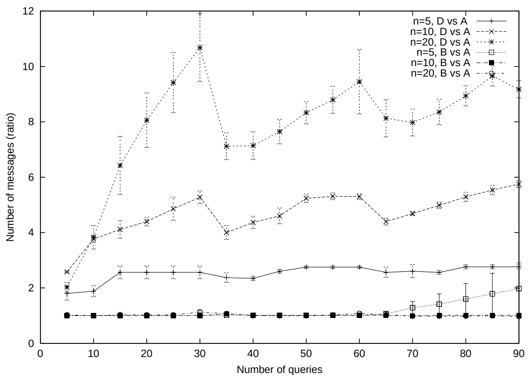

Figure 6. Ratio of number of messages

The graph shows in numbers what was explained in words in Section 4. Layering SD on top of IP in such environments can prove very costly. It can be several factors costlier in the number of messages when an application is service-discovery intensive - as many applications in these networks will be.

The graph shows no difference between cases (A) and (B) in terms of the number of messages sent since both of them integrate the SD layer with the IP layer. The small variations from the ratio of 1 (after about 60 queries) are seen because the SD information timeouts happened at different times during the experimental runs for cases (A) and (B). This happened since we did not have an event-driven simulation with tight control over time.

In the case of the comparison of case (D) with case (A), the ratio of the number of messages goes down each time there is a timeout of information at the nodes (since this would trigger full-floods once again even in case (A)).

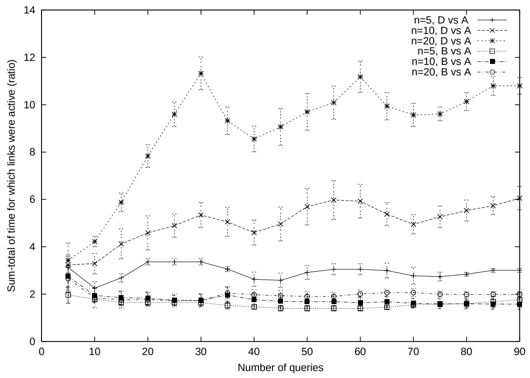

Figure 7. Ratio of link active mode time

Figure 7 shows the ratios for the other metric - the amount of time for which a link is kept active. These are for the same set of runs as for the previous graph. The graph shows the same comparison for cases (B) and (D), both relative to the integrated approach (A). Again the ratio dips occur during SD information timeouts (when case (A) also floods the network). When the IP and SD layers are also separate (case (D)), the performance is worse than with case (B) since there are more wasteful broadcasts (the point that was made in Section 4.3). The initial dip in the graphs in some cases is due to the fact that link-timeouts have not happened yet in the layered case; that is, there is no additional overhead of link-formation during this time in the layered case.

Again, we find that the lack of information across layers could lead to sub-optimal power usage. To realize link-mode power savings, information from the SD layer is crucial. This optimization is only one of the many possible with an integrated approach. Given that our preliminary studies show significant savings (a factor of 2 or more) and that power savings for low-end devices in these scenarios are crucial, cross-layer optimizations are desirable.

It is worth noting that just the reduction of floods by optimizing across the SD and IP layers brings about a lot of benefits. Further optimizations across the link-formation layer can result in more intelligent link-formation algorithms.

Varying other parameters

We examine the variation of two other parameters - the number of services, and the density of the graph. In both cases, the results exhibit the same trend although the exact numbers are different. An integrated approach has significant gains with respect to one metric or the other. For these cases, due to lack of space, we show only the graphs for the ratio of the time spent in active mode.

Figure 8 examines the effect of having more than one service in the network (with $n = 10$). With more number of services, the overhead of a layered approach is smaller. This is understandable since with more services, the integrated case would also take a lot of floods to find their locations.

Figure 9 shows the effect of having a denser or sparser configuration of nodes (with $n = 10$ & $S = 4$). We try the two extreme cases of a tree and a complete graph (denoted g=tree and g=nc2 in Figure 9). The overhead of layering in terms of the link efficiency is worse for denser graphs since more wasteful links are formed.

Figure 10 shows the real crux of the problem. In all the earlier cases, all the nodes in the network were equally active. We changed this to have half the nodes "inactive". That is, they don't have any services themselves that are accessed by others, and they themselves do not access other services. Ideally, they should participate in the network only if absolutely required – only when required to act as routers. An integrated approach allows such an optimal performance. With a separate link layer, these idle nodes would also form unnecessary links.

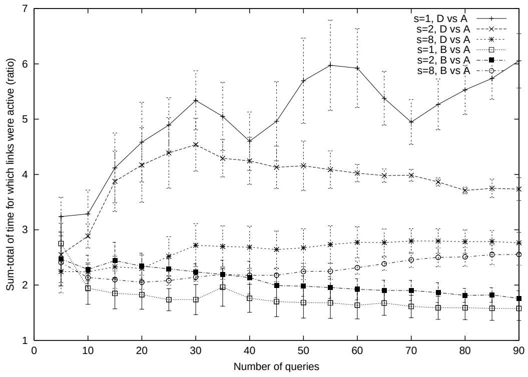

Figure 8. Ratio of link active mode time (S>1)

The graph shows the numbers only for the nodes that were idle (and not for the entire network). The overhead in terms of the amount of time for which these nodes have to keep their links active is over an order of magnitude. (The drop after about 20 queries is once again due to a timeout of information at nodes).

In this context, consider the power consumption

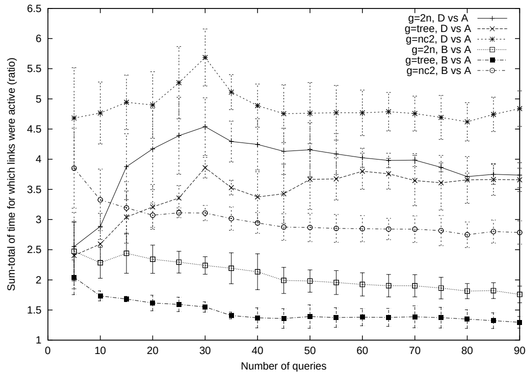

Figure 9. Ratio of link active mode time: with denser/sparser networks

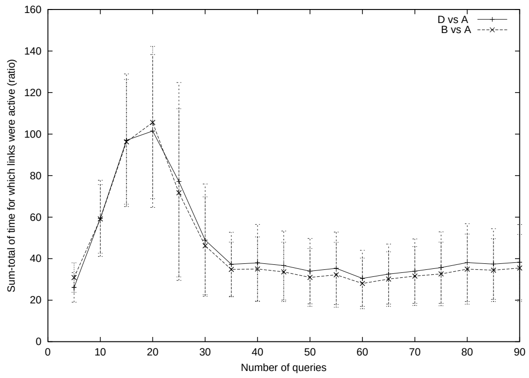

Figure 10. Ratio of link active mode time: with only half the nodes active

numbers from Cambridge Silicon Radio chip-sets (www.cambridgesiliconradio.com): (a) $40mA@2.8V$ in active mode – link established and ready to exchange data, and (b) $120\mu A@2.8V$ in park mode – link established, but not ready to transmit. Since power consumption in active mode is three orders of magnitude higher than in park mode – it is very important to manage transition between the two intelligently.

6. Discussion & Conclusions

Throughout the evolution of the TCP/IP suite of protocols demand for higher bandwidth at lower cost has fueled the development of new link technologies: Ethernet, ATM, SONET, DSL, wireless and cable networks, etc.. IP has been remarkably successful as an internetworking protocol; virtually all link media support IP today. Since Bluetooth is a new link technology, a standard for IP over Bluetooth is likely to evolve in the near future.

Bluetooth scatternets could bring about a new class of devices and services. To encourage novel applications in this domain, it is necessary to solve a variety of problems.

Bluetooth is a connection-oriented technology and scatternets are multiple-hop networks. With power considerations added in, these characteristics necessitate a rethinking of protocol layers. Cross-layer optimizations are not only possible, they are also natural due to the same scope of operation of the different layers and the on-demand operation.

The key is to preserve idleness. That is, a node wakes up and participates in a protocol only when absolutely essential. Battery-power should be used only for useful work, and not for unnecessary flooding or link maintenance. These devices could be very light-weight and designed for batteries to last for weeks, months, or even years. Enabling applications and new devices in this domain requires power optimizations in every possible way.

On-demand behavior is crucial to optimal operation of networks that have only brief periods of activity, or those that have only a portion of the network active at any time. An analogy worth noting is that of multicast in the Internet – sparse-mode PIM with on-demand *join* messages is preferred to dense-mode PIM with periodic *prunes* – when only a portion of the network is active, or when there exist large periods of inactivity [11].

The optimizations we have suggested are generic and could be used by all applications since service-discovery and the layers below are common to all applications. Further integration of the application layer also may bring in better performance, but such an optimization would be very application specific.

The service information caching feature of our integrated layer is similar to the promiscuous route learning feature in AODV. It also resembles *inference* of application level names in INS [6]. When a node learns how to contact another, all the intermediate nodes can also learn how to reach either of those nodes.

A possible concern related to caching service descriptions is in the context of "confidential service discovery" – a service might want only authenticated clients to discover its existence. In such cases, intermediates nodes would not be allowed to see or cache plain-text service descriptions. The benefits of caching would not apply to these services.

Finally, the cache of service information in an integrated approach can be conceptually thought of as a distributed implementation of an SLP Directory-Agent. The information is collected on-demand and cached.

Although our discussion has been centered around Bluetooth, the optimizations presented in this paper could be applied to similar technology in the future – the reason why existing solutions in other domains do not apply to scatternets is that the lower layer in a scatternet cannot operate efficiently without directives from the layers above.

Acknowledgements

We thank Steve Czerwinski, Ben Zhao, Charles Perkins and the anonymous reviewers for their feedback on earlier revisions of this paper.

References

[1] Bluetooth. http://www.bluetooth.com/.

[2] IETF Working Group, Zero Configuration Networking. http://www.ietf.org/html.charters/zeroconf-charter.html.

[3] Sun Microsystems, Jini Connection Technology. http://www.sun.com/jini/.

[4] *Bluetooth Baseband Specification, Version 1.0B*, chapter Part B, 10.8. http://www.bluetooth.com/, 1999.

[5] *Service Discovery Protocol, Bluetooth Specification, Version 1.0B*, chapter Part E. http://www.bluetooth.com/, 1999.

[6] W. Adjie-Winoto and et.al. The Design and Implementation of an Intentional Naming System. In *Proc. 17th ACM SOSP*, Jan 2000.

[7] AT&T. Virtual Network Computing. http://www.uk.research.att.com/vnc/.

[8] H. Balakrishnan, S.Seshan, and R. H. Katz. Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks. *ACM Wireless Networks*, Dec 1995.

[9] J. Broch, D. B. Johnson, and D. A. Maltz. *Dynamic Source Routing, Internet-Draft, draft-ietf-manet-dsr-00.txt, Work in Progress*, March 1998.

[10] S. Czerwinski and et.al. An Architecture for a Secure Service Discovery Service. In *MobiCom*, Aug 1999.

[11] S. Deering and et.al. An Architecture for Wide-Area Multicast Routing. In *Proceedings of SIGCOMM '94*, Sep 1994.

[12] Y. Y. Goland and et.al. *Simple Service Discovery Protocol/1.0, Internet Draft, draft-cai-ssdp-v1-03.txt, Work in Progress*, Oct 1999.

[13] E. Guttman and et.al. *Service Location Protocol, Version 2, Request for Comments: 2608*, June 1999.

[14] T. D. Hodes and et.al. Composable Ad hoc Mobile Services for Universal Interaction. In *MobiCom*, Sep 1997.

[15] M. Laubach and J. Halpern. *Classical IP and ARP over ATM, Request for Comments: 2225*, April 1998.

[16] V. D. Park and M. S. Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *INFOCOM*, April 1997.

[17] C. E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Computer Communication Review*. Oct 1994.

[18] C. E. Perkins and E. M. Royer. Ad-hoc On-Demand Distance-Vector Routing. In *2nd IEEE Wksp. Mobile Comp. Sys. and Apps.*, February 1999.

[19] C. K. Toh. Associativity-Based Routing for Ad-Hoc Networks. *Wireless Personal Communications*, Mar 1997.

[20] R. Troll. *Automatically Choosing an IP Address in an Ad-Hoc IPv4 Network, Internet-Draft, Work in Progress.*, Oct 1999.

[21] M. Weiser. The Computer for the Twenty-First Century. *Scientific American*, Sep 1991.