Capacity of Bluetooth Scatternets

CS 499 - B. Tech Project: Semester II, 2001-2002
Computer Science and Engineering Department
Indian Institute of Technology Kanpur, India - 208 016.

Supervisor: Dr. Dheeraj Sanghi
DUGC Convenor: Dr. Ajai Jain
Committee Members: Dr. Rajat Moona, Dr. Deepak Gupta
Project Group: Aaditeshwar Seth (98001), Anand Kashyap (98047)

Abstract

Most local area wireless technologies like IEEE 802.11 do not impose any restrictions on the topology of nodes and connections on which they will operate, but Bluetooth imposes some constraints for constructing valid topologies. The performance of Bluetooth over networks of a large number of nodes depends heavily on this topology, and in this paper we have evaluated the criteria for making optimal topologies which maximize the total throughput that can be achieved in a Bluetooth network. We then propose the Dynamic Scatternet Construction Protocol (DSCP) which constructs topologies on the fly, and specify routing schemes to work over the topologies constructed by DSCP.

As a supplementary work, we have also compared 802.11 and Bluetooth in the small distance Personal Area Networking scenario, and we find that Bluetooth holds a promising future in this arena since it has a greater degree of scalability in crowded networks of small area. We have also proposed a synchronization protocol which can bring about a desired synchronized pattern along multi-hop routes of data transfers. This synchronization will be important in the efficient utilization of the offered capacity.

Keywords: Bluetooth, Piconets, Scatternets, Bluescat, NS-2, DSCP

1 Introduction

Bluetooth [1] is a short range wireless technology intended to replace the cable(s) connecting portable and fixed devices. It operates in the unlicensed ISM band at 2.4 GHz, and uses a frequency hopping TDD (Time Division Duplex) scheme for transmission. The maximum bandwidth possible is 1 Mbps. On the channel, each packet is transmitted on a different hop frequency.

The Bluetooth system provides a point-to-point connection (using two Bluetooth devices), or a point-to-multipoint connection (using a maximum of eight Bluetooth devices). Two or more devices sharing the same channel form a piconet, with one device acting as the master of the piconet, and the other device(s) acting as slaves. At the most seven active slaves can remain attached to a master at any instant. In all cases, the master controls the channel access.

Multiple piconets with overlapping coverage areas form a scatternet. Each piconet can only have a single master. However, slaves can participate in different piconets on a time-division

multiplex basis. In addition, a master in one piconet can be a slave in another piconet. Different piconets are not time or frequency synchronized, and each piconet has its own hopping channel. The hopping sequence of frequencies in a piconet is a function of MAC address of the master. Therefore, no two piconets can have the same hopping sequence. The presence of scatternets becomes imperative when the number of active Bluetooth devices exceeds eight, and some slaves/masters have to act as bridging units to link the different piconets together. The bridge nodes are in HOLD mode in one piconet and CONNECTION mode in the other piconet. They switch between their two nodes in both piconets at the same time.

There are multiple ways of scatternet topology construction on the same ad hoc collection of nodes [6], and the overall network performance is greatly effected by the scatternet topology. For example, dispersed topologies with fewer number of bridges can lead to bottleneck bridges. Similarly, dense topologies with a large number of bridges can lead to wastage of bandwidth capacity of the bridges. Therefore, topology contraction becomes an orthogonal research issue. BTCP [5] was proposed as a scatternet topology construction protocol which connects nodes starting at the same time in a strongly connected network. However, in real life scenarios, nodes will hardly ever be switched ON at the same time. Therefore a dynamic topology construction scheme is needed which makes provision for realtime addition and deletion of nodes. In this paper we have proposed DSCP (Dynamic Scatternet Construction Protocol) which maximizes the overall network capacity in dynamic scenarios.

In section 2 we have described the simulation tool which we have used [4]. We then describe our experiments and results on optimal topologies maximizing the total network throughput in Section 3, followed by a description of the DSCP protocol in Section 4. Finally we present our conclusions and describe the work that we are currently pursuing as an extension to our experiments. We describe some additional work that we have done on improving scatternet capacity in Appendix-A and Appendix-B.

2 Simulation Tool

We have used the NS-2 simulator [2] and IBM's Bluehoc [3] patch for Bluetooth in NS-2, for a realistic modeling of the physical layer and the frequency hopping scheme followed by Bluetooth. The Bluehoc simulator originally had a very minimalistic support for scatternets, and hence we made the appropriate extensions to it described in detail in [4]. We then used this extended simulation tool for analyzing our theories and proposals. The simulator now models the Bluetooth stack according to the Bluetooth specifications, and allows the construction of both predetermined scatternet topologies, as well as topologies constructed on the fly. It uses the Deficit Round Robin (DRR) scheme [14] for the master coordinated Baseband scheduling, and the Best-Fit SAR policy [7] [8] [9] at the L2CAP layer. A limited version of the Scatternet Synchronization Protocol (described in Appendix-A) is used for synchronizing the devices along the scatternet chains. For analysis purposes, we set up varying traffic on different scatternet topologies, and monitor the performance by analyzing the generated trace files by Perl scripts.

For further simulations we are currently implementing DSCP in NS-2. This will help us identify and resolve important performance issues in our protocol.

3 Topology of Bluetooth scatternets

We have considered one main performance metric for comparing the performance of different scatternet topologies:

End-to-End Throughput: This metric provides a good estimate of the capacity of a scatternet by determining the amount of traffic that can flow in a given topology. When compared with the maximum capacity available, it also gives an estimate of the capacity utilization in the scatternet and the quality of the topology.

3.1 All nodes within range of each other

We begin by approximating the maximum attainable capacity in a Bluetooth scatternet where all nodes are within range of each other. The analysis is as follows:

A:	Area of the network
N:	Number if nodes in the network (uniformly distributed)
R:	Average distance between nodes $= \sqrt{\frac{A}{N}}$
P:	Average number of nodes in a piconet
$\frac{N}{P}$:	Number of piconets
L:	Average path length
$\frac{L}{R}$:	Average number of hops
$\frac{L}{R} * \frac{1}{2}$:	Number of masters in a path
F:	Number of flows
C:	Capacity at a master = 1 Mbps
$\frac{C}{P-1}$:	Bandwidth available to each link
$\frac{C}{2(P-1)}$:	Per flow throughput through a master

Therefore,
F* $\frac{L}{2R} \sim \frac{N}{P}$
$\Rightarrow$ F $\sim 2(\frac{N}{P})$ / $\frac{L}{R}$

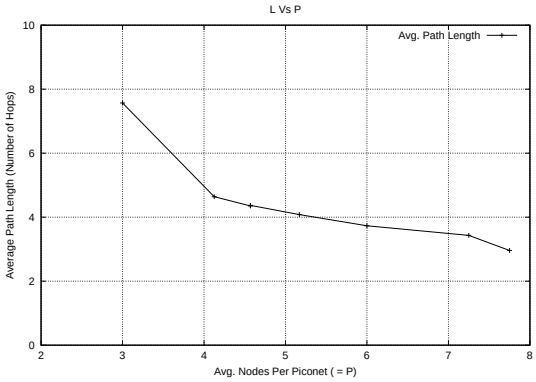

Figure 1: L Vs P for 25 nodes

$\Rightarrow$ Network capacity = F * $\frac{C}{2(N-1)} \sim (\frac{N}{P} * \frac{C}{(P-1)}) / \frac{L}{R}$

$$Capacity \sim \frac{C * \sqrt{AN}}{P(P-1)L} \quad (1)$$

This indicates that the scatternet capacity varies inversely with the number of nodes in a piconet (= P) and with the average path length (= L). This is also expected because lesser number of nodes in a piconet implies greater number of piconets in the network, and hence an increased parallelism in the network. Similarly, lower average path lengths imply more capacity because longer chains waste the capacity at all the intermediate masters. However, the interesting part is that L is inversely dependent to a large extent on P, as shown in Fig. 1.

A larger number of piconets (lesser P) generally results in longer average path lengths. An average P=8 implies saturated piconets, and may result in several bottlenecks being created in the network. An average P=3 results in a topology of a linear chain, and thus increases the average path length to a very large quantity. This clearly gives us reason to expect that an optimal P might lie some where in between.

It is clear that L also depends on the number of bridges in a piconet - that is, L depends on the connectivity of the piconet with other piconets.

A well connected piconet will result in a smaller L, even if P is large. To determine an optimal number of bridges in a piconet, we again reflect on the basic purpose of a bridge. If the bridge has to transmit data to one piconet only, then it might as well revert back to its status of a slave. If it remains a bridge, it implies that it is involved in forwarding the traffic of some other device. Furthermore, it is obvious that the number of slaves in a piconet should be comparable to the number of bridges. A very small number of bridges results in the bridge becoming a bottleneck if all the slaves need to transmit through that bridge. A very large number of bridges results in capacity wastage, since the bridge does not serve its purpose fully. An optimal load distribution results when the number of slaves is equal to the number of bridges, or very close to it.

Therefore:

$$Capacity \propto \frac{1}{p} \tag{2}$$

$$Capacity \propto \frac{1}{L} \tag{3}$$

$$L = f(P, Slave : Bridge, Connectivity) \tag{4}$$

To verify our analysis, we conducted two sets of experiments keeping the number of nodes in the network as fixed at 25 for the first set, and at 35 for the second. We were not able to go beyond 35 nodes because the simulator was unable to handle a larger network. We ran all simulations for 60 simulated seconds, and averaged our results over five simulation runs on the same topology under different traffic scenarios. This gives us a good estimate of the metrics achievable even in practical situations.

In order to exhaustively analyze all topologies to select the most optimal one, we manually laid out scenarios as a function of three parameters - the number of nodes in a scatternet, the average number of bridges in a piconet, and the average number of slaves in a piconet. Having a uniform policy for analyzing topologies helps us in generalizing the results, which is otherwise very difficult because of the large number of possible combinations. The policy is as follows:

N: Number of nodes
P: Targeted average number of nodes in a piconet
S: Average number of pure slaves in a piconet
B: Average number of bridges in a piconet
n: Number of piconets

1. n=$\frac{N}{1+S+\frac{B}{2}}$
2. Make n piconets
3. **if** $B == 2$ **then**
4. Connect piconets in a ring
5. **if** $B == 3$ **then**
6. Connect piconets in a ring
7. Connect alternate nodes in groups of 3 consecutive nodes
8. **if** $B == 4$ **then**
9. Connect piconets in a ring
10. Connect alternate nodes
11. Equally distribute remaining devices as pure slaves in all piconets

It is not necessary to analyze topologies beyond the ones enumerated by the above policy because those other scenarios will be a combination of two or more of these analyzed topologies.

Given the total number of nodes, we begin by trying to achieve the targeted P. Since, N = Number of masters (= n) + Number of pure slaves (= n * S) + Number of bridges (= $n/2$ * B), hence we can calculate the number of piconets required to construct the desired topology. Since topologies cannot be laid out with all piconets achieving the exact desired P, therefore we alter the number of slaves in each piconet,

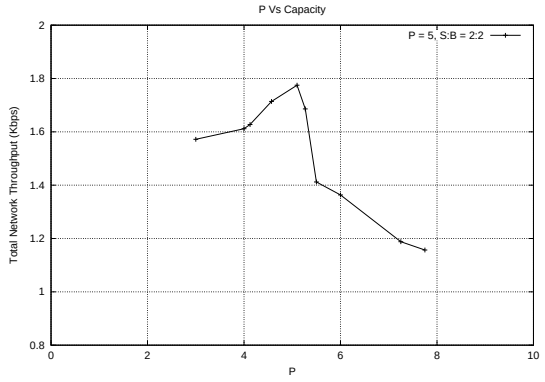

Figure 2: Capacity Vs P for 25 nodes

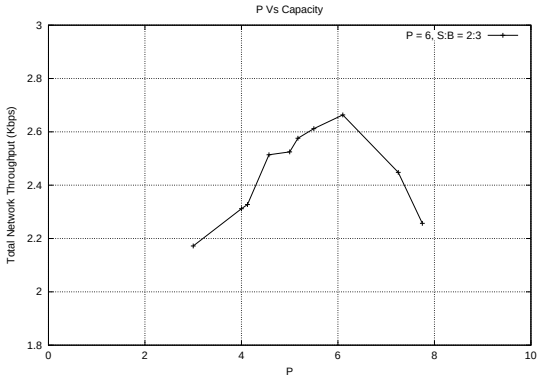

Figure 3: Capacity Vs P for 35 nodes

but not the targeted number of bridges. This is because the performance of each piconet will depend on the bottlenecks created by the number of bridges connected to the master. Therefore, the topologies constructed are in the form of a ring, with possible interconnections if B is greater than *2*.

On these topologies, we then laid out the maximum possible traffic by randomly selecting pairs of nodes, and carried out simulations on our simulation tool. Our results are shown in Fig. 2 for 25 nodes, and Fig. 3 for 35 nodes. Based on these results, we can safely conclude that maximum capacity is obtained in topologies with $P = 5$, $S = 2$, $B = 2$ for 25 nodes, and $P = 6$, $S = 2$, $B = 3$ for 35 nodes. Clearly, the transition from $P = 5$ to $P = 6$ occurs because as the ring gets larger, the average path-length increases, thus decreasing the overall network capacity.

We can see from the graphs that the maximum capacity for 25 nodes connected with P = 5 and S:B = 2:2, is less than that for 35 nodes connected with P = 6, and S:B = 2:3. Also, it seems to be logically correct that a network of 35 nodes with P=6 and S:B = 3:2 will be more optimal than a network of P=5 and S:B = 2:2. However, it is interesting to compare these networks with a scenario where there are two clusters of P = 5 and S:B = 2:2 instead of a single ring. We compare these different cases in Fig. 4 by simulating scenarios of the number of nodes increasing to 35. For simplicity, we assume no inter-cluster traffic in the case with two clusters. Based on the simulations, we can conclude that multiple clusters turn out to be better than single ring networks for up to 35 nodes. The question of choosing an appropriate cluster size remains, and is explored in the next subsection.

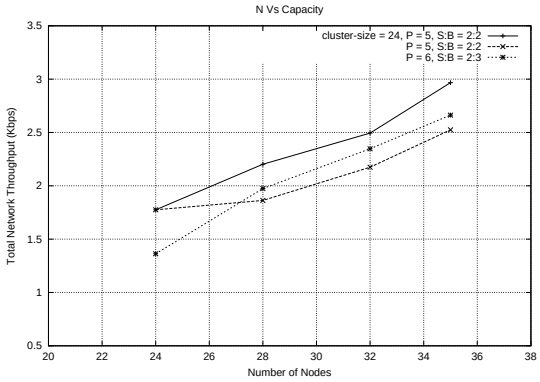

Figure 4: Capacity Vs N: Cluster partitioning

3.2 All nodes not within range of each other

We now extend our analysis to larger scatternets where all nodes might not be within range of each other. We proceed by constructing small clusters of nodes with all nodes within a cluster being in range of each other. Inter cluster

communication can be done by periphery nodes of adjoining clusters. The cluster size is an important factor in this case and we did many experiments, both theoretically and by simulations to come up with the best cluster size, ie. which gives maximum capacity.

3.2.1 Theoretical analysis

The simulation tool cannot handle very large scatternets (larger than 35 nodes) and hence we did a theoretical analysis to find the maximum network throughput for the scatternet. We did the analysis by varying the cluster size from 16 to 28 (16, 20, 24, 28). For each cluster size, we find the capacity successively for an increasing number of nodes. It is assumed that each cluster can be reached from every other cluster in atmost one hop, which will most probably be true for most practical cases also. For calculation purposes, we completely differentiate the cases of intra-cluster and inter-cluster communication, ie. we assume that all the clusters together are either communicating within themselves or across themselves. For this purpose, we keep a factor λ ($0 < \lambda < 1$) which indicates the ratio of intracluster traffic to the total traffic. We then calculate the overall capacity as λ * intercluster traffic + (1-λ) * intracluster traffic. The intracluster capacity is calculated by simply multiplying the number of clusters with the capacity for a single cluster obtained through the earlier simulation results. To calculate the intercluster throughput, we let every cluster form one single link with every other cluster in the network. This gives a total of ${}^nC_2$ links where n = number of clusters, and assuming that all these links will be completely loaded, the total throughput in this case comes out to be ${}^nC_2$ * 361 Kbps.

We did the analysis by keeping the values of λ as 0.7, 0.8, and 0.9 because we assume that the nodes will be communicating with the nodes within the cluster for more time than the nodes outside the cluster. Fig. 5, Fig. 6 and Fig. 7 show graphs for the capacity against the number of nodes, for $\lambda = 0.7$, 0.8 and 0.9 respectively. In each graph, there are 4 lines for the cluster sizes of 16, 20, 24 and 28. It is evident from the graph that the cluster size of 20 gives the maximum capacity. The result can be explained intuitively by observing that as the cluster size increases, the intracluster throughput increases because average path length within a cluster does not increase by as much as what the ring-size increases, thus leading to more parallel communications. On the other hand, the intercluster throughput decreases with increase in cluster size because the number of clusters decreases. Therefore there is a trade off and the best result is obtained at a cluster size of 20.

3.2.2 Simulation analysis

The result obtained above was also verified by simulations done for the number of nodes up to 35. The perfect synchronization between clusters as assumed in the theoretical analysis is not possible in the actual simulation. Therefore to assume a λ value (say 0.9), we run 10 simulations and make scenarios such that each node remains on a connection within its cluster for 9 runs, and on a connection across clusters for 1 run. Similarly, if we only need to run 5 simulations, then for clusters of size 20, we select 10 nodes that remain within their cluster for all 5 runs, and the other 10 nodes are kept within the cluster for 4 runs and across clusters for 1 run. The results obtained for $\lambda = 0.9$ are shown in Fig. 8 and they are in close similarity to the values obtained theoretically. This verifies the theoretical result of the best cluster size coming out to be 20.

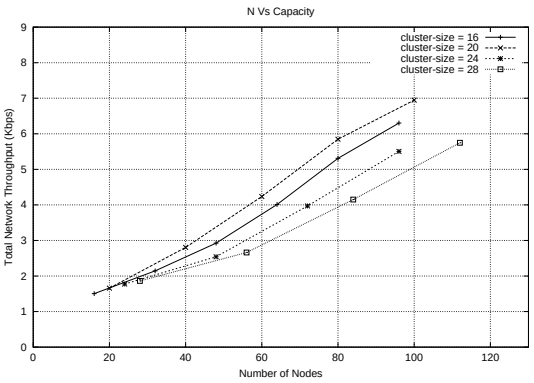

Figure 5: Capacity Vs Number of Nodes: Theoretical, $\lambda = 0.7$

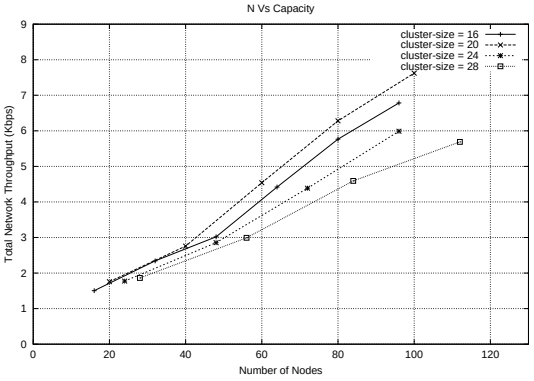

Figure 6: Capacity Vs Number of Nodes: Theoretical, $\lambda = 0.8$

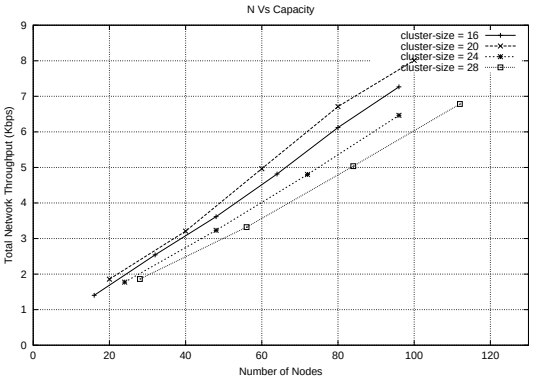

Figure 7: Capacity Vs Number of Nodes: Theoretical, $\lambda = 0.9$

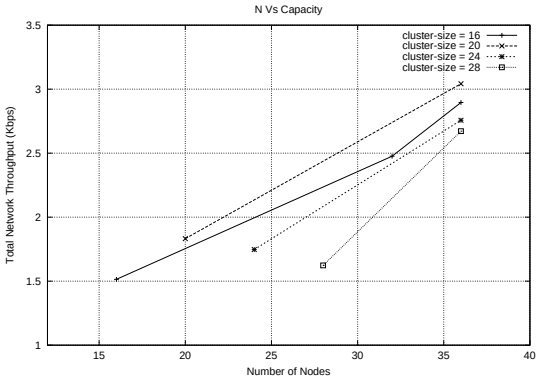

Figure 8: Capacity Vs Number of Nodes: Simulation, $\lambda = 0.9$

4 DSCP

We model DSCP (Dynamic Scatternet Construction Protocol) on the same pattern as ZRP (Zone Routing Protocol) [16] for ad hoc networks. ZRP makes mutually exclusive zones of a certain radius and follows a table-driven protocol like DSDV [17] within a zone, and an on-demand protocol like AODV [15] for communication across zones. Similarly, for routing we follow a table-driven protocol within each cluster, and an on-demand protocol for communication across clusters.

4.1 Communication within clusters

Topology construction for this part has two functionalities - addition of nodes and removal of nodes. Finally, a table-driven routing protocol will be needed for setting up and maintaining connections on the topology of the lcuster constructed. Addition of nodes simply implies including new nodes into the existing ring. When the ring becomes full, it is expanded by making the device the master of a new empty piconet, and thus introducing further vacancies in the ring. Similarly, removal of nodes implies the systematic deletion of nodes from the ring, followed by shrinking the ring when a piconet be-

comes empty. The process is facilitated by all the masters being made aware of the clocks of all the other masters, and of any topology change as soon as the rearrangement concludes.

4.1.1 Addition of nodes

Whenever a new node enters a network having more than one cluster in its vicinity, it has the option of entering any cluster. However, the best performance can be derived by completing a ring of 5 piconets with a cluster size of 20 nodes, as shown in the previous section. Hence the node will have to continue listening for replies from masters of more clusters for INITIAL_WAIT_TIMEOUT time units. Whenever it connects to a master, it is communicated the number of devices already present in that cluster. After expiry of the INITIAL_WAIT_TIMEOUT, it finally connects to the cluster having the maximum number of nodes.

The rearrangement process to be commenced on the attachment of a new device to some master in a scatternet is delayed by a random backoff greater than MAX_PROPAGATION_TIME in the network. This is done to prevent any conflicts that might arise from the addition of two nodes in quick succession. If the master receives any rearrangement messages during this time, then it relapses again into a repeat random back-off.

After the backoff, the master now determines which vacant piconet will accommodate the new node, and communicates the arrangement to the new device as well as broadcasts it to the rest of the masters in the scatternet. The broadcast is restricted in a manner similar to the caching procedure of the <BROADCAST_ID, BD_ADDR> tuple, as followed in the AODV ad-hoc routing protocol. The master also sends an FHS packet of the destination master to the new device, in order to prevent a repeat INQUIRY-PAGE procedure. The new node can thus attach itself to the selected vacant piconet.

If a vacant piconet is not available, then expansion of the ring becomes necessary. Since the master might be attached to two or more piconets, it now disconnects itself from the adjoining piconet having the least master BD_ADDR. This disconnection request is also broadcasted over the scatternet, and to the new device in the same way as explained above. The new device is thus made the master of a new piconet and included in between the two piconets.

4.1.2 Removal of nodes

There are three cases when a node removes itself from the scatternet - either it is was a pure slave, or a bridge, or a master.

Pure slave: Any ring shrinkage produces at least two surplus devices - a master and an associated bridge - in addition to any attached slaves. Clearly, a ring can be allowed to shrink only if there are enough vacancies present in the scatternet and these are consolidated within the same piconet. Hence, proceeding from a full ring, one slave is allowed to depart. When a second slave departs, the scatternet is rearranged to consolidate the two vacancies into a single piconet. A third slave is also allowed to depart from some other piconet. When a fourth slave departs, the scatternet is rearranged to consolidate the two new vacancies into a single piconet. Then the ring is shrunk by dissolving the piconet with the master having the lower BD_ADDR, and the free master and bridge are accommodated into the vacant piconet.

Bridge: Whenever a bridge node departs, at least one of the two adjoining piconets becomes non-vacant. The scatternet is rearranged by removing a pure slave from an adjoining piconet, and replacing it as the new bridge. If this rearrangement makes both the adjoining piconets

vacant, then rearrangement is done as in the previous case of the departure of a fourth slave.

Master: When a master of a non-vacant piconet departs, then a pure slave can be rearranged as the new master. However, if the master of a completely vacant piconet departs, then an adjoining non-vacant piconet can always be selected. Rearrangement is done by relocating a pure slave from the adjoining piconet, and making it the new master. The vacancy produced in the adjoining piconet can be dealt with as in the previous case of removal of a pure slave.

4.1.3 Routing

Intra-cluster routing is table-driven.

Route discovery: All masters communicate the BD_ADDRs of their respective slaves and bridges and the population of their neighbors to their immediate neighbors. The masters receiving this information update their tables and calculate the number of hops to each device. In this way, even in a ring of a maximum size of five piconets, all the nodes can be made aware of the shortest routes to all other nodes in that cluster. Thus, no formal route discovery procedure is needed as such, but only a table lookup is required.

Route maintenance: Any addition or removal of nodes is communicated immediately to all the nodes in that cluster. The broadcasts are restricted as described before. All nodes, including the source node, update their tables immediately on receiving this information. Since the ring structure is always maintained, hence a route to the destination nodes can always be established again.

4.1.4 Intra-cluster communication

Since we follow a table-driven protocol within a cluster, hence source-routing of data packets is not required. Data transfer can be done simply by looking up the shortest route to a destination and MAC-unicasting along the correct route.

4.2 Communication across clusters

This proceeds in two stages - cluster discovery and inter-cluster communication. Cluster discovery is done periodically throughout the lifetime of the scatternet. An on-demand protocol is used for route discovery and route maintenance for establishing and sustaining inter-cluster communication.

4.2.1 Cluster discovery

Each cluster should be connected directly to as many clusters as possible. This ensures single-cluster-hop paths to the maximum clusters possible, and hence overall shortest paths too. We target at allocating a pair of nodes for each pair of clusters directly connected to each other. This is done in the following way: Whenever each pure slave of a cluster is not involved actively in any connections, it disconnects from its parent cluster for a period of CLUSTER_DISCOVERY_TIMEOUT duration and enters into an INQUIRY - INQUIRY_SCAN phase. Whenever this slave establishes a connection with another slave of a different cluster also in CLUSTER_DISCOVERY phase, it exchanges a synchronization timer of ROUTE_QUERY_TIMEOUT duration, and both the slaves return to their respective clusters. Thenceforth, out of these two slaves the one which was in INQUIRY phase and hence was the slave of the temporary piconet of these two nodes, is delegated as the link-node for that pair of clusters. For the other cluster, the master of the slave which had temporarily disconnected, is

delegated as the link-node. After this, the slave link-node disconnects from its parent cluster every ROUTE_QUERY_TIMEOUT, and connects with the corresponding master link-node. This connection is used for periodic information and query exchange, which is used during the route discovery and route maintenance phases.

4.2.2 Routing

Routing is done in an on-demand manner, and comprises of route-discovery and route-maintenance.

Route discovery: Whenever a route request is made for a node outside a cluster to which a route is not already known, the request is broadcasted to all the nodes of that cluster. Out of all these receiving nodes, whenever the link-nodes which had been delegated as the cluster representatives during the cluster discovery phase connect with each other, they propagate the route request from the querying cluster to the next cluster. Since all the nodes of a cluster are aware of all the members of their respective clusters, hence they immediately respond to the query if they contain the destination node, or know of a route to the destination node. Else the route request is broadcasted to all other link-nodes of this cluster, and subsequently to the clusters beyond this. The route discovery now works in a manner similar to the AODV protocol, with the link-nodes of each cluster maintaining reverse links and finally selecting the path with the smallest number of cluster-hops. Broadcasts are prevented from exploding by caching the <BROADCAST_ID, BD_ADDR> tuple.

Route maintenance: Whenever a node removes itself from a cluster and a link breaks, the information is immediately communicated to all the nodes of that cluster and the ring is reconfigured. Therefore, whenever a non link-node or a non destination-node breaks away, the ring is reconfigured and the traffic can be recommenced. However, if a destination node breaks away, then the downstream link-node of that cluster unicast this update to the source, which terminates its traffic. Similarly, if a link-node breaks away, then the previous downstream link-node will not be able to transmit any data, and after a LINK_NODE_RELAXATION_TIMEOUT it unicasts a ROUTE_ERR to the source. The source will now do another route discovery and only then recommence its traffic.

4.2.3 Inter-cluster communication

After a route discovery has been completed, inter-cluster communication becomes possible by table-driven MAC-unicasting within a cluster from the incoming-link-node to the outgoing-link-node. Similarly, the link-nodes transfer data to their corresponding counterparts in the next cluster by MAC-unicasting in the standard way. In both cases, all nodes along any route are aware of their next-hop nodes, and hence source routing is not needed.

5 Conclusions

We derived that the capacity will depend on the average number of nodes in a piconet, and the average path-length. The path-length in turn is dependent on the number of nodes per piconet, and the slave:bridge ratio in the piconets. Through simulations and analysis, we found that the most optimal capacities result with scatternets having clusters of 20 nodes with 5 piconets, each piconet having an average of 5 nodes and an equal number of slaves and bridges. We then described the Dynamic Scatternet Construction Protocol, which constructs topologies on the fly based on the optimization results derived through our experiments, and takes care of routing through a scheme similar to the ZRP protocol for ad hoc

networks.

As a supplementary work, we also found that Bluetooth has a lot to benefit from the parallel communications that are possible in networks having multiple piconets. This increases the scope for comparing Bluetooth with IEEE 802.11b, and evaluating various factors which determine their relative performances in different application scenarios. We saw that Bluetooth has a very bright future in giving a cheap, yet robust infrastructure for developing Personal Area Networking applications.

We also saw that the best deliverable capacities of scatternets relies considerably on synchronization along the transfer chains, and quality of the topology constructed. For synchronization, we proposed a simple master-coordinated synchronization correcting protocol based on the traffic-type, and verified a simplified version of the protocol.

6 Current work

We are presently implementing DSCP in NS-2 for further consolidating our protocol and evaluating parameters for optimizing the performance of the protocol.

7 Acknowledgments

We want to convey our sincerest acknowledgments to Dr. Rajeev Shorey from the IBM India Research Lab. at New Delhi, for introducing us to this problem, and giving us valuable guidance throughout the year on the topic.

References

[1] Bluetooth SIG. Bluetooth Specifications. In *http://www.bluetooth.com*

[2] Network Simulator (NS), version 2.1b7a manual. In *http://www.isi.edu/nsnam*

[3] IBM. Bluehoc Simulator. In *http://oss.software.ibm.com/developerworks/opensource/bluehoc*

[4] Aaditeshwar Seth, Anand Kashyap, Dheeraj Sanghi. A Simulation Tool for Bluetooth Scatternets: Extensions to Bluehoc. *Submitted for publication.*

[5] Theodoros Saloidis, Pravin Bhagwat, Leandros Tassiulas, Richard LaMaire. Distributed Topology Construction of Bluetooth Personal Area Networks. In *IEEE INFOCOM, 2001.*

[6] Pravin Bhagwat, Srinivasa Rao. On the characterization of Bluetooth Scatternet Topologies. In *http://www.cs.umd.edu/ pravin/bluetooth*

[7] Abhishek Das, Abhishek Ghosh, Ashu Razdan, Huzur Saran, Rajeev Shorey. Enhancing Performance of Asynchronous Data Traffic over Bluetooth Wireless AdHoc Networks. In *IEEE INFOCOM, 2001*

[8] Aaditeshwar Seth, Anand Kashyap, R K Ghosh. Performance Study of Bluetooth over High Bandwidth Applications. In *Proc. of IEEE ADCOM, 2001*

[9] Manish Kalia, Deepak Bansal, Rajeev Shorey. MAC Scheduling and SAR Policies for Bluetooth: A Master Driven TDD Pico-Cellular Wireless System. In *IEEE MOMUC 1999.*

[10] Jinyang Li, Charles Blake, Douglas S, J De Couto, Hu Imm Lee, Robert Morris. Capacity of Ad Hoc Wireless Networks. In *MOBICOM, 2001.*

[11] Pravin Bhagwat, Adrain Segall. A Routing Vector Method (RVM) for Routing in Bluetooth Scatternets. In *IEEE MOMUC 1999.*

[12] Bhaskaran Raman, Pravin Bhagwat, Srinivasan Seshan. Arguments for Cross Layer Optimizations in Bluetooth Scatternets. In *Proc. of SAINT, 2001.*

[13] A Proposed Flow Specification: RFC-1363. IETF (Internet Engineering Task Force. In *http://www.ietf.org/rfc/rfc1363.txt*

[14] M. Shreedhar, G. Varghese. Efficient Fair Queuing Using Deficit Round Robin. In *ACM SIGCOMM, 1995.*

[15] Ad Hoc On Demand Distance Vector (AODV) protocol: IETF Draft. IETF (Internet Engineering Task Force. In *http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-10.txt*

[16] Zone Routing Protocol (ZRP) protocol: IETF Draft. IETF (Internet Engineering Task Force. In *http://www.cs.hut.fi/ mart/mail/manet/ 1997/0089.html*

[17] Charles E Perkins and Pravin Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proceedings of the SIGCOMM '94.*

Appendix A: Scatternet Synchronization Protocol (SSP)

Considering the case when data has to be transfered across several piconets over a chain of alternate masters and bridges, the best throughput will be achieved when the alternate bridges connect on disjoint intervals of time with their common master. However, with disjoint intervals the end-to-end delays will be proportional to (chain_length - 2). If delays are to be minimized, then the intervals should overlap exactly, but in this case the effective flow throughput will be halved. Hence, the synchronization policy developed should be dependent on the kind of traffic, with the first policy of maximum throughput being used for high bandwidth applications like FTP, and the second policy of minimum delays being used for delay sensitive applications like streaming audio.

Even if the desired synchronization is somehow attained, the bridges tend to get out of sync. with the masters. This is because whenever a bridge wants to switch from one piconet to the other, it first sends a HOLD request to the current master, and waits for the HOLD acknowledgement. This introduces delays at two levels:

- When the bridge is waiting for a POLL from the current master, so that it can send its HOLD request to the master.
- When the master receives the HOLD request, but delays its acknowledgement because of the scheduling policy that it is following.

These delays can be expected to remain constant when the traffic at the masters is not too heavy, but will fluctuate heavily when the traffic load is increased and the scheduling policies are not able to scale.

The initial synchronization can be achieved when the connection is established and the QoS negotiated through any of the SDP protocols. However, the subsequent operational protocol should be correcting in nature, since bridges tend to get out of synchronization often. If it is assumed that no intermediate nodes will have any self-traffic, but will only be concerned with forwarding the traffic, then this leads to a simple correcting algorithm to be followed by the masters along the chain. The algorithm described below, is for maximizing the throughput. An equivalent algorithm can be developed for minimizing the delays, depending on the traffic type.

Variables -

source_bride: bridge bringing traffic into master
dest_bridge: bridge carrying traffic away from master
conn_time[ ]: times for which bridges remain connected with master (negotiated during connection establishment)
curr_time[ ]: time for which slave has been connected with master most recently

Procedure -

1. **When** *source_bridge* connects, **then begin**
2. **If** *dest_bridge* connected, **then begin**
3. Send **HOLD** command to *dest_bridge* for *conn_time[source_bridge]*
   end
   end
4. **When** *dest_bridge* connects, **then begin**
5. **If** *source_bridge* connected, **then begin**
6. Send **HOLD** command to *dest_bridge* for (*conn_time[source_bridge]* - *curr_time[source_bridge]*)
   end
   end

For modifying the algorithm for smaller delays, the master has to only delay the acknowledgement of the HOLD request of the destination bridge, and synchronize it with the time when the source bridge exits the piconet.

The above algorithm will work even in the case when self-traffic (traffic originated by the bridges themselves, or by the other slaves or masters) is allowed, but only when such traffic is installed before the forwarding traffic is set up. Otherwise, any new traffic additions will change the bandwidth allocation at the master, and hence change the amount of time that the bridges need to remain connected with the master. Any change of this connection time will require the development of a separate protocol to communicate the change to the bridges. To avoid such a protocol, we suggest an alternate algorithm which is slightly biased towards forwarding traffic, but we expect that over a period of time this forwarding traffic will terminate, and the differentials will average out.

Assuming that the scheduling policy implemented at the masters will be maintaining a list of preference ratios in which the bandwidth will be allocated to the different slaves/bridges at any instant of time, our objective is just to reset the original bandwidth ratio to a forwarding traffic whenever it connects. It is safe to generalize that any self traffic from/to a bridge can be considered to be equivalent to a self-traffic from/to any other pure slave. Therefore:

Variables -

bw_ratio[ ]: band-width allocation ratios
self_ratio: ratio of bw allocated to forwarding traffic of bridge

Procedure -

1. **Whenever** *bridge* connects for first time for forwarding traffic, **then begin**
2. *self_ratio* = *bw_ratio[traffic]*
   end
3. **Whenever** *bridge* connects subsequently for forwarding traffic **OR** *bridge* is connected **AND** master reallocates bw, **then begin**
4. *diff_ratio* = *bw_ratio[traffic]* - *self_ratio*
5. distribute *diff_ratio* among each non-forwarding traffic
   end
6. **Whenever** *bridge* disconnects after first time from forwarding traffic, **then begin**
   restore original ratios
   end

We implemented the SSP scheme in the simulator, assuming that all traffic is set up initially and started together. Therefore, we did not have to reallocate the bandwidths when a forwarding traffic was about to be recommenced. For veri-

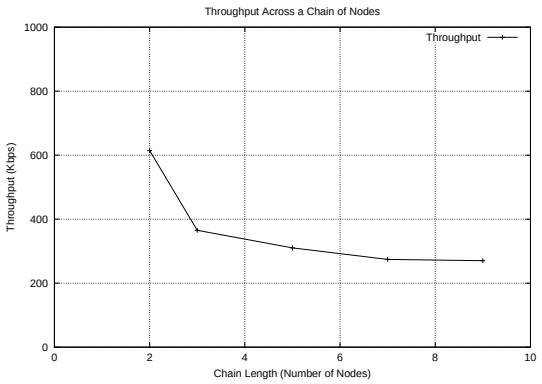

Figure 9: Capacity Across a Chain

fying our protocol, we constructed topologies of linear chains of two to nine nodes, and forwarded the maximum traffic possible. We scheduled the bridges to alternate between the two piconets, remaining for 640 slots in each.

The results of the maximum flows we obtained are shown in Fig. 5. For a trivial piconet system of one master connected to one slave, a maximum flow of 600 Kbps is attained. We notice that this is less than the theoretical maximum of 723.2 Kbps because the master temporarily detaches itself from the slave to go into an INQUIRY procedure to capture more slaves. As the chain length increases, the throughput drops from 360 Kbps at a chain length of three, to saturate at 275 Kbps for longer chains. The end-to-end delays obtained are also in close agreement with the theoretically expected delays of ((chain_length - 2) * presence_time). Perpetual synchronization is not achieved because of the delays encountered at two levels, when the bridge sends a HOLD request to its current master. These delays are of the order of 20 - 40 slots, and disturb the synchronization.

Appendix B: Comparison of 802.11 and Bluetooth

Before presenting any comparisons, it is important to recognize the application area in which the comparisons should be conducted. IEEE 802.11 offers a bandwidth of 2Mbps over a range of 250m, while 802.11b offers a bandwidth of 11 Mbps. When bandwidth loss in MAC layer headers and MAC interactions of collisions of RTS packets, etc is taken into account, the deliverable capacity reduces, but still remains at 1.7 Mbps and 9.4 Mbps respectively. This makes the technology ideal for establishment of long-distance ad hoc or infrastructure networks, as well as for usage in small distance scenarios. Bluetooth on the other hand, is functional over ranges of 15m, although higher range Bluetooth chips are being made now. The maximum bandwidth available is 1 Mbps, which reduces to an effective 723.2 Kbps for asymmetric communication. Therefore, Bluetooth can only be applied essentially for indoor networking over small distances.

However, even though Bluetooth has a lower bandwidth, it has an advantage that it does not use a common channel for communication, but each piconet uses a different channel. Therefore, the available bandwidth in a scatternet becomes equals to the combined capacity of the number of piconets in the network, with traffic flowing in all the piconets simultaneously. 802.11 on the other hand uses a single channel, and in small distance scenarios when all nodes are within hearing distance of each other, the CSMA protocol of 802.11 reduces the available bandwidth to that of just one 802.11 device. This behavior of 802.11 was researched in [10], and the capacity was observed to drop significantly as the number of nodes was increased, due to an increased contention at the MAC level in the network.

Clearly, Bluetooth appears to be more scalable than 802.11 in small distance application sce-

narios. However, the lower bandwidth of Bluetooth does not leave much scope for competition, except for the lower cost of the Bluetooth chips (almost 20 times less than 802.11b wireless cards). But all is not lost for Bluetooth. The limit of 1 Mbps on Bluetooth was placed because of the FCC ruling that frequency hopping systems could not hop at a rate exceeding 1 Mhz. This limited the bandwidth to 2 Mbps. However, very recently the FCC relaxed its ruling and the present limit is now at 10 MHz, thus bring the deliverable bandwidth to 10 Mbps. The costs of the frequency hopping sub-system will increase if this extra bandwidth is to be accessed, but it will narrow the gap between Bluetooth and 802.11b, and increase competition.

We have simulated both the 2 Mbps IEEE 802.11 WLAN and Bluetooth in an area of 15m X 15m, while increasing the number of nodes from two to twenty. We have used the BTCP protocol to construct scatternet topologies in the Bluetooth case, and studied the two technologies by pumping traffic from the masters to all the slaves in their respective piconets. We have then used the same pair of devices to study the traffic limitations in the 802.11 case. The results clearly demonstrate that Bluetooth scales much better than 802.11 when all devices in the network are within each others communication range. BTCP constructs the minimum number of piconets, with each piconet connected to every other piconet in the scatternet. Therefore, when the total number of devices in the scatternet is up to eight, the effective capacity delivered is that of a single master. When the number of nodes are increased, this capacity follows a stepped function, inflating to the combined capacity of two masters, and then to the capacity of three masters. On the contrary, 802.11 also moves in a stepped manner, but the total network throughput decreases as the number of nodes are increased.

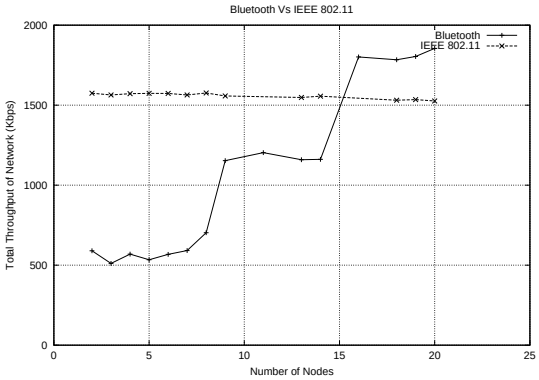

Figure 10: IEEE 802.11 Vs Bluetooth

Bluetooth does not attain the theoretical maximums of 723.2 Kbps, 723.2 * 2 Kbps, and 723.2 * 3 Kbps, because the masters spend some time in the INQUIRY state, listening for more slaves. Even 802.11 does not attain the maximum of 1.7 Mbps because the MAC layer contention and collision of RTS packets, limits the available capacity.

As is observed, Bluetooth overtakes the 2 Mbps 802.11 WLAN when the number of devices in the area increases beyond fifteen.