Cross-Layer-Based Adaptive Traffic Control Protocol for Bluetooth Wireless Networks

Sabeen Tahir
Department of Information Technology,
Faculty of Computing and Information Technology,
King Abdulaziz University, Jeddah 21589, Makkah,
Saudi Arabia

Sheikh Tahir Bakhsh
Department of Computer Science,
Faculty of Computing and Information Technology,
King Abdulaziz University, Jeddah 21589, Makkah,
Saudi Arabia

***Abstract*—Bluetooth technology is particularly designed for a wireless personal area network that is low cost and less energy consuming. Efficient transmission between different Bluetooth nodes depends on network formation. An inefficient Bluetooth topology may create a bottleneck and a delay in the network when data is routed. To overcome the congestion problem of Bluetooth networks, a Cross-layer-based Adaptive Traffic Control (CATC) protocol is proposed in this paper. The proposed protocol is working on backup device utilization and network restructuring. The proposed CATC is divided into two parts; the first part is based on intra-piconet traffic control, while the second part is based on inter-piconet traffic control. The proposed CATC protocol controls the traffic load on the master node by network restructuring and the traffic load of the bridge node by activating a Fall-Back Bridge (FBB). During the piconet restructuring, the CATC performs the Piconet Formation within Piconet (PFP) and Scatternet Formation within Piconet (SFP). The PFP reconstructs a new piconet in the same piconet for the devices which are directly within the radio range of each other. The SFP reconstructs the scatternet within the same piconet if the nodes are not within the radio range. Simulation results are proof that the proposed CATC improves the overall performance and reduces control overhead in a Bluetooth network.**

***Keywords*—Bluetooth; scatternet; multi-layer; resolving bottleneck; reducing control overhead component**

I. Introduction

Improvements in wireless technologies have enhanced our daily life. A number of mobile devices can interconnect through wireless technology and exchange different types of data (text, voice, and video) [1]. Bluetooth is an open standard that has the ability to connect heterogeneous mobile devices. A basic communication unit of Bluetooth is piconet, which consists of eight active Bluetooth nodes. A piconet is created through sharing the same frequency hopping sequence and synchronization, where one Bluetooth node becomes the master and remaining nodes act as slaves. An active Bluetooth device may perform the role of master, slave or bridge. Slave nodes cannot communicate directly with each other; they always need a master node support for communication, as the master node always handles all communications within a piconet [2], [3]. The communication within a piconet is also called intra-piconet communication. Bluetooth devices transmit their data packets over Time Division Duplex (TDD) [4].

Bluetooth also allows communication within multiple piconets, which is known as a scatternet. Where a relay or bridge device provides communication among different piconets, a bridge node can be Master-Slave (M/S) or Slave-Slave (S/S) [5]. A bridge node is responsible for transporting messages between piconets so that the resources should not be restricted [6]. Bluetooth efficient communication can be achieved through a role switching technique [7], which can be used for different requirements. A role switching operation divides one piconet into multiple piconets; splitting operation increases the number of piconets and bridge nodes. Using an example in Fig. 1(a), before executing a splitting role switch operation, there is one piconet having one master with six slave nodes. Fig. 1(b) shows how a role switch operation splits one piconet into two piconets P_1 and P_2 by changing the roles of the devices, where node C and F perform the master role and node A is used as a bridge between two piconets.

A merge role switch operation combines different piconets into a single piconet [8], [9]. As shown in Fig. 1(b), two piconets (P_1 and P_2) are connected through an intermediate node *A*. Nodes (*B*, *C*, *D*, *E*, *F*, *G*) are in the range of node A. According to the role switch operation, node A performs the master role and merges two piconets into a single piconet. Fig. 1(a) shows how a bridge node becomes a master and masters (*C* and *F*) change their roles from master to slave. A role switch operation can be applied on Bluetooth device to take over the resource of other device. During this operation, devices can change their roles from slave to master and vice versa. As shown in Fig. 1(c), node *D* and *G* become masters and node *A* acts as a slave node, with two independent piconets.

Many researchers proposed scatternet formation protocols [10]-[14] to decrease scatternet formation time or increase the probability of making a scatternet, but an efficient scatternet formation protocol is missing. This paper designs a well-organized protocol for a Bluetooth scatternet that minimizes the delay and efficiently uses the network resources. The proposed CATC controls and shares the traffic load of master and bridge nodes in a distributed manner. The role-switching operations are performed dynamically for congestion handling on an affected link.

Deanship of Scientific Research (DSR) in King Abdulaziz University, under grant No. (611-271-D1435).

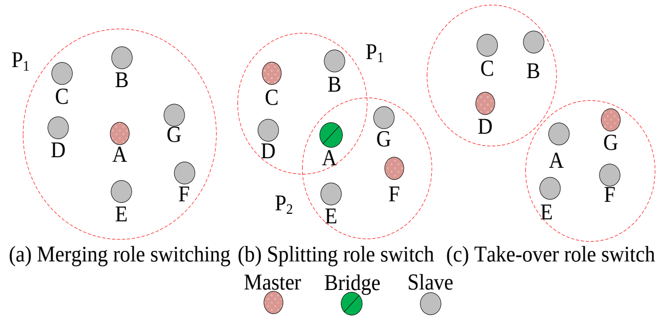
P1
C
B
D
A
G
E
F
C
B
P1
D
A
G
P2
E
F
C
B
D
A
G
E
F
(a) Merging role switching
(b) Splitting role switch
(c) Take-over role switch
Master
Bridge
Slave

Fig. 1. Different configurations of Bluetooth network.

The paper is organized as: The background is discussed in Section 2. To control traffic load in Bluetooth network, a protocol is proposed in Section 3. The results of the proposed CATC are presented in Section 4 using NS-2 [15] and UCBT [16]. The paper is concluded in Section 5.

II. Related Work

The traffic bottleneck is an important issue in Bluetooth, which is caused by a master or bridge node. Within a piconet, all slaves communication is possible through the master node, while scatter communication is achieved through intermediate relay node [17]. A huge number of devices may cause congestion and delay in the Bluetooth scatternet. The slave device cannot communicate with each other, master is always involved in intra-piconet communication among slaves. Therefore, master's energy and mobility have a critical role in the piconet. In the same way, bridge node mobility and energy has a crucial role for inter-piconet communication. Failure of a bridge node may disconnect the whole network. Many researchers have proposed different techniques for a Bluetooth scatternet, i.e., relay optimization, congestion avoidance, and scheduling. Each technique has its own benefits and limitations [1], [18], [19]. Through a literature survey, some relevant existing research works have been analyzed in this research work.

Dynamic piconet restructuring protocol (PRP) [20] is proposed for Bluetooth networks. PRP locally regulates the traffic on the master node. PRP shares master node load by forming new piconets of slave nodes that can communicate directly, where one slave node acts as a master and others act as slaves. During the restructuring operation, the slave node with light traffic flow will be selected as the new master. For example, as shown in Fig. 2(a), nodes (*A*, *B*), (*E*, *F*) and (*B*, *C*) are communicating through a master *G*. According to PRP, when traffic load is detected by the master node, it performs piconet restructuring using a role switching operation as shown in Fig. 2(b).

It is analyzed that PRP provides a solution for congestion problem on a master node through sharing the load, but it creates serious problems. It loses active member addresses due to breaking the existing link between slaves and master. During transmission, if new joins the piconet, the master assigns all remaining active member addresses to new nodes. Once the communication is over the nodes cannot join the existing master due to unavailability of active member address. Frequent piconets construction also consume extra resources. At it creates new piconets for all communicating pairs without considering whether they are frequently communicating or not. The new nodes cannot communicate with the node already changed their state, therefore, the new nodes have to wait until nodes to return to their original states.

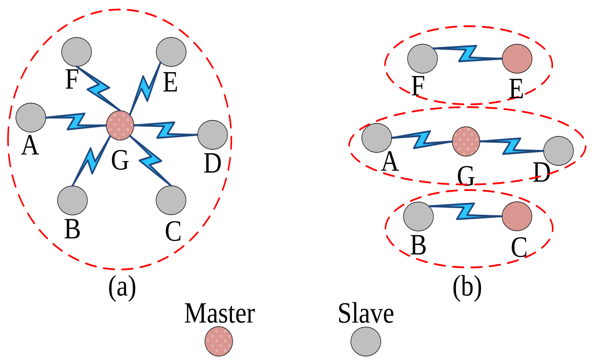

Fig. 2. Traffic flow analysis before and after role switching.

Subsequently, Dynamic Congestion Control (DCC) [21] has been proposed as another solution for avoiding bottleneck problems in a scatternet through backup relay (BR). If several links use a single bridge it creates a bottleneck. The master monitors the load and delay on the bridge node. The master gets relay load and a number of links from relay table. When the master observes bottleneck in the piconet, it shares the load through BR. As shown in Fig. 3, multiple links passing through M_1 the data traffic load can be determined by the master. As different piconets are communicating through bridge node B_1, the master activates the *BR* to share the load. It provides a solution for intra-piconet congestion inter-piconet congestion it still missing. When there is a bottleneck on B_1, but the master in P_1 cannot find the congestion due to distributed traffic, it fails to avoid inter-piconet congestion. As shown in Fig. 3, all traffic load is passing through B_1. Although *BR* is available the load is distributed, *BR* is not activated by M_1. In distributed load, DCC does not allow parallel transmissions.

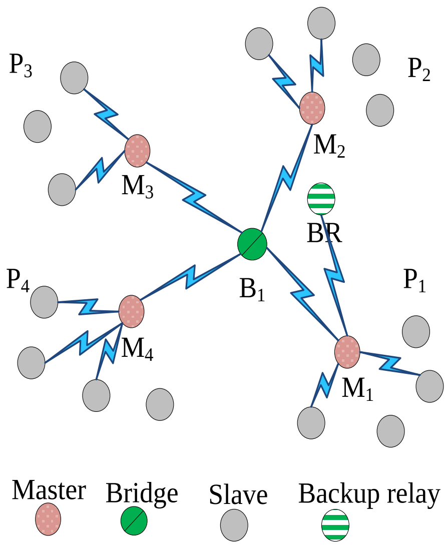

Fig. 3. Scatternet formation using DCC Protocol.

III. The Proposed Cross-Layer-Based Adaptive Traffic Control (CATC) Protocol

This section discusses the proposed CATC protocol for intra-piconet and inter-piconet congestion avoidance. The proposed protocol consists of two parts; in the first part, role switching techniques are used to overcome the problem of intra-piconet congestion avoidance. In the second part, FBB is used to control the bridge load that overcomes the bottleneck problem of inter-piconet congestion.

A. *Intra-piconet traffic load and dynamic role switching operation*

In this section, intra-piconet traffic load handling is presented. The intra-piconet traffic load is handled through PFP and SFP.

1) Intra-piconet load handling through Piconet Formation within Piconet

Large numbers of connections passing through the master node within a piconet may create congestion. The incoming data traffic is called Download Traffic (*DTr*), and the outgoing data traffic is called Upload Traffic (*UTr*). The traffic load in a piconet is calculated as follows:

$$DTr = \sum_{i=1}^{n} a_{i1} \quad for\ i = 1,2,3,\dots n \quad where\ n \leq 8 \qquad (1)$$

$$UTr = \sum_{j=1}^{n} a_{1j} \quad for\ j = 1,2,3,\dots n \quad where\ n \leq 8 \qquad (2)$$

$$TT = DTr + UTr = \sum_{i=1}^{n} a_{i1} + \sum_{j=1}^{n} a_{1j} \qquad (3)$$

where *DTr* and *UTr* calculate incoming and outgoing traffic respectively. The total traffic (*TT*) load on a master node is calculated through the sum of (1) and (2).

The proposed protocol maintains a Master Traffic Flow Table (*MTFT*) to monitor traffic load. The *MTFT* maintains the information of all incoming and outgoing data traffic going through the master within a piconet as recorded in Table 1. A threshold (θ) value is used for congestion handling on the master node. When a master gets *TT* it compares to θ, where θ = 90 slots. The traffic load is calculated after receiving or transmitting data between a new pair. In the next step, master marks the most frequent (*MF*) communicating nodes that reach the limit of the threshold value. Hence, the CATC performs network restructuring using a taking-over role switching operation. When the master node determines higher traffic load is greater than θ, it performs a role switch operation by sending a request packet to the pair of *MF* communicating nodes within the piconet. When the master node receives uplink data, it checks in the *MTFT*; if the number of active connections is more than three, the master node calculates *TT*. A pair of nodes having the highest traffic load is marked as the *MF* communicating pair.

The role switch request packet contains the node *ID* and clock-offset of the nodes. On receiving the role switch request, the source node enters into Page state and destination node enters into Page Scan state to create a new piconet. In the next step, the master node changes both nodes mode into park mode, to save active member addresses and reduce unnecessary switching control overhead. Once the communication ends, the nodes come back into their original states and send a request to the master node to restore their original states as active slaves. As the master node maintains the list of nodes for temporary connections, the original active member addresses are reserved by the master node. Hence, the nodes can come back to their original states without losing their connection.

Using an example, Fig. 4(a) shows, node (*I*, *J*) and (*A*, *B*) are marked as *MF* communicating pairs by M_1 and M_2 respectively. Therefore, a piconet restructuring request is sent to the slave nodes by the master nodes. As a result, after piconet restructuring, slave nodes *J* and *A* become auxiliary masters; the new connections of the most frequently communicating nodes are shown in Fig. 4(b). The data traffic flow of *the* M_2 is maintained in Table 1 among different nodes, where *MF* represents a heavy traffic flow, 1 is used for normal traffic, and Ø means there is no data exchange between nodes.

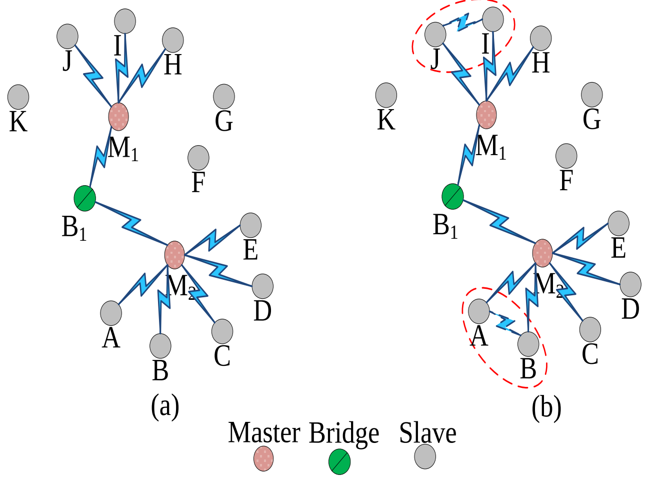

Fig. 4. (a) Before role switching operation (b) After role switching operation.

TABLE. I. Data Traffic Flow Analysis on Master M_2

ID	A	B	C	D	E	F	B_1	M_2
A	Ø	MF	Ø	Ø	Ø	Ø	Ø	1
B	MF	Ø	1	Ø	Ø	Ø	Ø	1
C	Ø	Ø	Ø	1	1	Ø	Ø	1
D	Ø	Ø	1	Ø	1	Ø	Ø	1
E	Ø	Ø	1	1	Ø	Ø	Ø	1
F	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
B_1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
M_2	1	1	1	1	1	Ø	1	Ø

2) Intra-piconet traffic load handling through Scatternet Formation within Piconet (SFP)

The proposed SFP creates a scatternet within a piconet. On receiving a role switch request, source and destination nodes enter Page and Page Scan state respectively and try to create a new link. As the paging procedure needs 1.28 *s*, after executing twice paging procedure if nodes fail to establish the new link. The source node sends a link fail message to the master node. On receiving the source node link fail message, the master requests connected slave nodes to enter into Inquiry state and create a new connection, where all slave nodes enter into the Inquiry Scan state and listen to the source and the destination nodes. A node that can connect both source and destination nodes performs a bridge role and executes a splitting role switching operation by making a scatternet within a piconet. During SFP operation, an intermediate node is selected as an auxiliary bridge (*AxB*) and a pair of source and destination nodes are selected as auxiliary masters. An intermediate node between source and destination can be selected as an auxiliary bridge.

The SFP operation is explained through Fig. 5. Nodes H and F are marked as MF communicating nodes in the piconet but both are not within the direct radio range of each other. Thus, node G performs an A x B role, while the source node H and the destination node F perform an *AxM* role. In Table 2, according to the Fig. 5, the master node updates the Node Information Table (*NIT*) for *MF* communicating nodes, which are in the same piconets but cannot communicate directly so they need an intermediate node.

TABLE. II. NODE INFORMATION TABLE (NIT) FOR P1 AFTER SFP

ID	Clock-offset	Device-role	Download traffic	Upload traffic
F	C-offset (F)	AxM	70	80
G	C-offset (G)	AxB	150	150
H	C-offset (H)	AxM	80	70

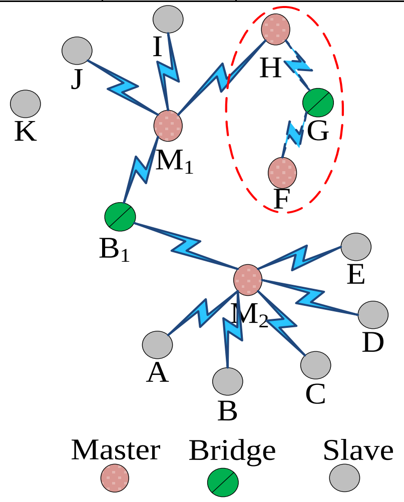

Fig. 5. Sharing of traffic load by making scatternet within the piconet.

B. Inter-piconet traffic load handling on the bridge node through Fall-Back Bridge (FBB)

When multiple piconets are connected through a single bridge, it may create a bottleneck in the network due to the unavailability of a bridge. The inter-piconet problem is solved through FBB. At the same time, a maximum of seven master devices can connect to a bridge device. According to the Bluetooth specification, a bridge node shares its time with all connected masters. Therefore, at the same time, only one master node's traffic can flow through the bridge node. Due to unavailability of a bridge node, inter-piconet congestion seriously affects network performance. The proposed CATC maintains a Bridge Traffic Flow Table (*BTFT*) (Table 3) to store the traffic load of masters that passes through the bridge node. As the bridge device receives/transmits data from master devices if a bridge device receives the data from a master device, it is called Bridge Download Traffic (*BDTr*); similarly, if the bridge device transmits data to the master device, it is called Bridge Upload Traffic (*BUTr*). The traffic load on a bridge (*CB*) device can be calculated as follows:

$$CB = BDTr + BUTr$$

$$= \sum_{i=1}^{n} a_{i1} + \sum_{j=1}^{n} a_{1j} \ \ for\ i\ \&j = 0, 1, .. n\ where\ n < 8 \qquad (4)$$

If higher traffic load is detected by a bridge node, it requests masters to activate a backup node. On receiving the request, master finds a FBB; if any master node has a FBB, then it sends a request to the bridge node to activate its connection with the required master node. The FBB is activated when a single bridge node is not sufficient for an efficient communication between piconets. Thus, parallel transmissions are allowed between piconets for well organized and smooth communication. Meanwhile, the master node sends the active bridge node into the park mode. As shown in Fig. 3(a), B_1 connects multiple nodes and creates a bottleneck. The heavy traffic flow does not allow parallel transmission in a scatternet, and thus, B_1 creates the bottleneck, as one master node sends data through B_1, and others wait for B_1 to become free. As shown in Fig. 6, node A is selected as FBB for P_2 and P_3 and node *D* is selected as FBB for P_1 and P_2. The dotted lines show the temporary links where traffic is shared and the bottleneck problem is resolved through activation of FBB. Master nodes update the *NIT* and send FBB to park mode; once communication ends successfully, FBBs return to their original states.

TABLE. III. BRIDGE TRAFFIC FLOW TABLE FOR B_1

ID	M_2	M_3	M_4	M_5	B_1
M_2	Ø	MF	1	1	1
M_3	MF	1	1	1	1
M_4	1	1	Ø	1	1
M_5	1	1	1	Ø	1
B_1	1	1	1	1	Ø

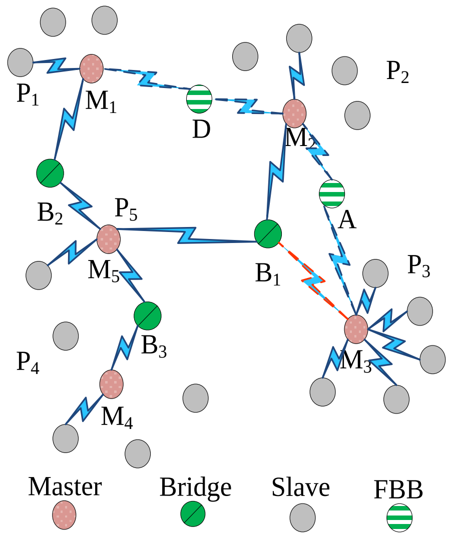

Fig. 6. Scatternet formation after activation of FBB.

IV. Performance Analysis

The proposed protocol is compared against PRP and DCC protocols. To assess the performance, the CATC is simulated in the University of Cincinnati Bluetooth (UCBT) [16], which is a ns-2 [15] based Bluetooth simulator. UCBT is an open source and publicly available which can support mesh-formed Bluetooth scatternet and implemented most Bluetooth protocol stacks [22]. The time interval between different frequencies is 625μs.

A. Simulation setup

The parameters used in the proposed protocol simulation are listed in Table 4. For simulation, the number of Bluetooth nodes is varied from 10 to 100 and 48 node pairs are used [23].

TABLE. IV. Simulation Parameters

Parameters	Assessment
Traffic Model	CBR
Number of nodes	10 - 100
Bluetooth nodes pairs	48
Simulation time	1000 s
Network Dimension	80 m x 80 m
Data packet type	DH3, DH5
Communication range	10 m
Scheduling algorithm	Round Robin
Bridge scheduling algorithm	Maximum Distance Rendezvous Point
Packet size	349 Bytes
Inquiry time	10.24 s
Paging time	128 – 256 s
Packet interval	0.15
Queue length	50 packets

B. Simulation results and discussion

In this sub-section, the simulation results are discussed. The simulation was run ten times and results are obtained using those ten simulations. After getting the comparison results, it was found that the proposed CATC protocol outperformed the existing PRP and DCC protocols.

During the communication, it was observed that, when the number of passing links increased through a single bridge node, the proposed CATC activates a FBB that shares the traffic load. The CATC allows the parallel transmission to reduce the wait time and improve network performance. In the holding mode when one pair of devices transmits remaining pairs are blocked. As DCC and PRP both are proposed to handle congestion, but, it is observed that the DCC is efficient for intra-piconet traffic load. When traffic load increases on a master node, it activates a bridge node for load balancing. In the contrary, the PRP solves bottleneck problem by creating extra piconets within the piconet. It creates a new piconet for each new communicating pairs. As shown in Fig. 7, the CATC shares the traffic load more efficiently compared to DCC and PRP. There are total twenty available bridges and 84 connections, the PRP uses 12 bridges, the DCC 14, and the CATC uses 18 bridge nodes. Therefore, the traffic load has been successfully shared; which improves the overall performance.

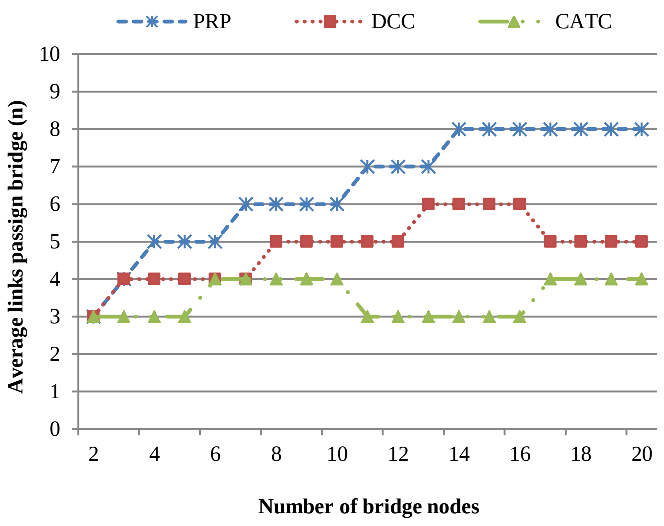

Fig. 7. Average links passing through bridge nodes vs. Number of bridges.

A large number of nodes in a scatternet increasing the master polling time. To increase efficiency, PRP frequently performs piconet restructuring for all connections. It provides the solution for congestion on the master node, but it increases the network delay. On the contrary, DCC avoids congestion, within an intra-piconet but it does not provide any solution inter-piconet. It is analyzed a large number of connections passing through a bridge node create a bottleneck and increase network delay. The CATC shares the traffic load on the master and bridge nodes. The total delay of protocols is shown in Fig. 8 and it is observed that CATC has less delay compared to PRP and DCC protocols. The throughput of the CATC, PRP, and DCC protocols was compared and it was observed that the CATC protocol showed better results than the PRP and DCC protocols. The PRP and DCC protocols consume more control packets compared to the CATC protocol. The CATC allows the parallel transmission because it efficiently manages traffic load in the intra-piconet and inter-piconet to improve overall network throughput. When a bridge creates bottleneck the CATC activates the FBB for an efficient communication between the piconets. As shown in Fig. 6, when a larger number of links pass through B_1 it creates a bottleneck, to avoid bottleneck node *A* is selected as a FBB for P_2 and P_3, and node *D* is selected as FBB for P_1 and P_2. Fig. 9 shows the throughput of the CATC is higher compared to PRP and DCC.

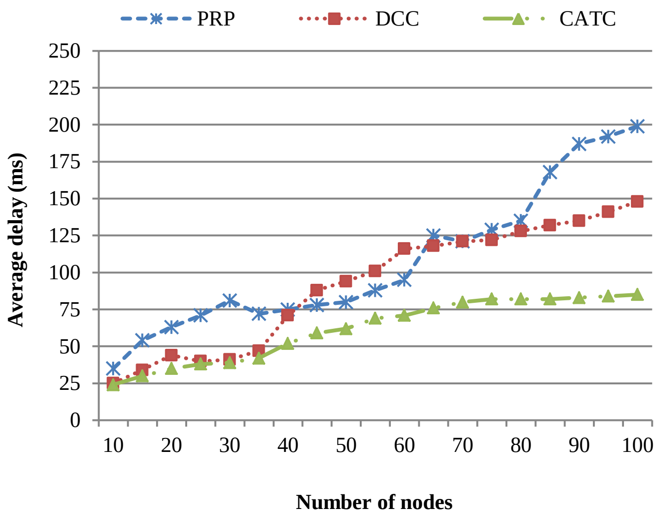

Fig. 8. Average delay vs. Number of nodes.

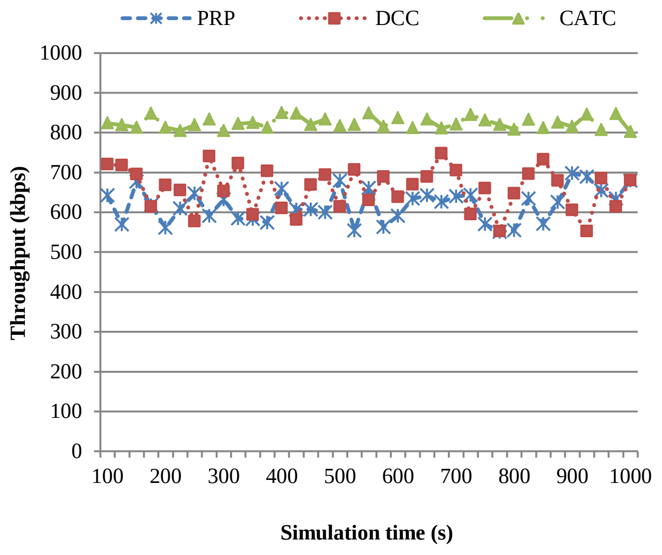

Fig. 9. Network throughput vs. Time.

Bluetooth has limited resources, and therefore, efficient resource utilization is key to network performance. The CATC does not frequently perform the network restructuring within the piconet, and therefore, it uses a lower number of control packets. The PRP frequently creates new piconets within the piconet and makes new links so that each time during synchronization, the Bluetooth devices use extra control packets. In contrast, the DCC protocol overcomes the delay problem within the piconet through activating a backup device that is utilizing extra control packets. Fig. 10 shows that the PRP and DCC's inefficient resource utilization causes more control packets compared to the CATC protocol. Also the number of blocking users increases due to the unavailability of intermediate nodes. As PRP makes new piconets frequently within the piconet and if other devices need to communicate with the devices which have changed their roles, it could block more users. The CATC protocol creates efficient links for intra-piconet and inter-piconet communication so it decreases the rate of blocking users. When the CATC protocol performs network restructuring, it changes the mode of the device in the park mode. After successful transmissions, it changes back into the original states. From Fig. 11, it can be seen that the CATC protocol performs better than the PRP and DCC protocol in terms of blocking connections about 15%.

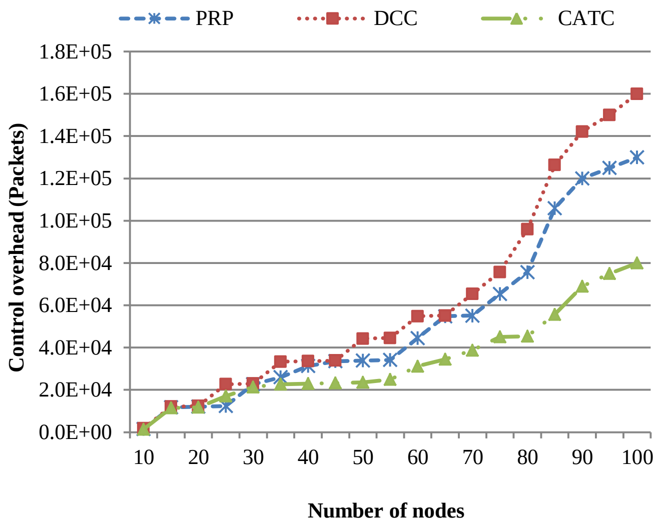

Fig. 10. Control packet overhead vs. Number of nodes.

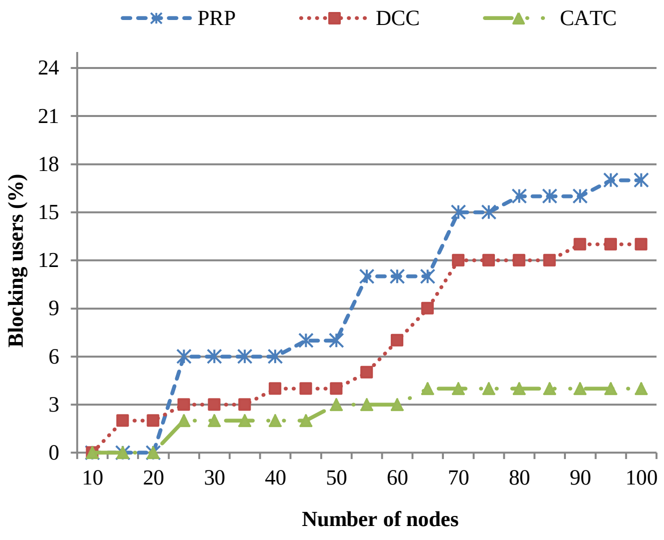

Fig. 11. Blocking users vs. Number of nodes.

V. Conclusions and Future Work

This paper has proposed a Cross-layer-based Adaptive Traffic Control protocol for Bluetooth network. The proposed CATC protocol shares the master load through network restructuring and the bridge load through FBB. The CATC creates PFP if nodes are within the range. If the source and destination are not within 10 m CATC creates SFP to reduce traffic load on the master node. On the contrary, the CATC activates FBB to overcome bottleneck problem of a bridge node and allow parallel transmission in the scatternet. Simulation results show that the CATC protocol outperforms existing protocols in terms of minimizing the total delay, control overhead, and a number of blocked users.

In future work, this research work will be extended by using some additional parameters for comparison. In addition, the network traffic load can be shared by reducing the hop count based on the role switch operations.

Acknowledgement

The authors are thankful to the Deanship of Scientific Research (DSR) at King Abdulaziz University for funding this project, under grant No. (611-271-D1435).

References

[1] M. J. Sataraddi, et al., “Priority Based Scheduler for Bluetooth Network,” in Advances in Computing, Communication, and Control. vol. 361, S. Unnikrishnan, et al., Eds., ed: Springer Berlin Heidelberg, 2013, pp. 356-365.

[2] Bluetooth_specificaiton. (2015). Bluetooth SIG, “Specification of the Bluetooth System”, http://www.bluetooth.com, June 2010.

[3] G. Aldabbagh, et al., “QoS-Aware Tethering in a Heterogeneous Wireless Network using LTE and TV White Spaces,” Computer Networks, vol. 81, pp. 136-146, 2015.

[4] Y. Chih-Min and Y. Yin-Bin, “Reconfigurable Algorithm for Bluetooth Sensor Networks,” Sensors Journal, IEEE, vol. 14, pp. 3506-3507, 2014.

[5] S. Sharafeddine, et al., “A scatternet formation algorithm for Bluetooth networks with a non-uniform distribution of devices,” Journal of Network and Computer Applications, vol. 35, pp. 644-656, 2012.

[6] J. Decuir, “Bluetooth Smart Support for 6LoBTLE: Applications and connection questions,” Consumer Electronics Magazine, IEEE, vol. 4, pp. 67-70, 2015.

[7] T. Klajbor, et al., “A new role-switching mechanism optimizing the coexistence of bluetooth and Wi-Fi networks,” Telecommunication Systems, pp. 1-11, 2010.

[8] G. Aldabbagh, et al., “Distributed dynamic load balancing in a heterogeneous network using LTE and TV white spaces,” Wireless Networks, pp. 1-12, 2015.

[9] S. T. Bakhsh, et al., “Self-Schedule and Self-Distributive MAC Scheduling Algorithms for Next-Generation Sensor Networks,” International Journal of Distributed Sensor Networks, 2015.

[10] C. M. Yu and J. H. Lin, “Enhanced bluetree: A mesh topology approach forming bluetooth scatternet,” Wireless Sensor Systems, IET, vol. 2, pp. 409-415, 2012.

[11] C. M. Yu, “Global configured method for blueweb routing protocol,” Communications, IET, vol. 6, pp. 69-75, 2012.

[12] J.-W. Lin and W.-S. Wang, “An efficient reconstruction approach for improving Bluetree scatternet formation in personal area networks,” Journal of Network and Computer Applications, vol. 33, pp. 141-155, 2010.

[13] G. Ramana Reddy, et al., “An efficient algorithm for scheduling in bluetooth piconets and scatternets,” Wireless Networks, vol. 16, pp. 1799-1816, 2010/10/01 2010.

[14] S. Bakhsh, “A Self-organizing Location and Mobility-Aware Route Optimization Protocol for Bluetooth Wireless,” Journal of King Saud University-Computer and Information Sciences, pp. 239-248, 2016.

[15] E. Hossain, “The Network Simulator (NS-2)”. http://www.isi.edu/nsnam/ns/ns-build.html, 2016.

[16] D. Agrawal and Q. Wang, “University of Cinicinnati Bluetooth simulator (UCBT)” http://www.cs.uc.edu/~cdmc/ucbt/, 2016.

[17] S. T. Bakhsh, et al., “Adaptive Sleep Efficient Hybrid Medium Access Control algorithm for next-generation wireless sensor networks,” EURASIP Journal on Wireless Communications and Networking, vol. 2017, pp. 84-94, 2017.

[18] P. A. Laharotte, et al., “Spatiotemporal Analysis of Bluetooth Data: Application to a Large Urban Network,” Intelligent Transportation Systems, IEEE Transactions on, vol. 16, pp. 1439-1448, 2015.

[19] J. Nieminen, et al., “Networking solutions for connecting bluetooth low energy enabled machines to the internet of things,” Network, IEEE, vol. 28, pp. 83-90, 2014.

[20] G.-J. Yu and C.-Y. Chang, “Congestion control of bluetooth radio system by piconet restructuring,” Journal of Network and Computer Applications, vol. 31, pp. 201-223, 2008.

[21] S. Tahir Bakhsh, et al., “Dynamic Congestion Control through backup relay in Bluetooth scatternet,” Journal of Network and Computer Applications, vol. 34, pp. 1252-1262, 2011.

[22] F. Subhan, et al., “Indoor positioning in bluetooth networks using fingerprinting and lateration approach,” in International Conference on Information Science and Applications (ICISA), pp. 1-9, 2011.

[23] P. Johansson, et al., “Rendezvous scheduling in Bluetooth scatternets,” presented at the IEEE International Conference on Communications, 2002.