DOI: 10.21917/ijct.2011.0058

HOLD MODE BASED DYNAMIC PRIORITY LOAD ADAPTIVE INTERPICONET SCHEDULING FOR BLUETOOTH SCATTERNETS

G.S. Mahalakshmi[1] and S. Senthilkumar[2]

Department of Computer Science and Engineering, Anna University, Chennai, India

E-mail: [1]mahalakshmi@cs.annauniv.edu and [2] senthil_ssk@hotmail.com

Abstract

Scheduling in piconets has emerged as a challenging research area. Interpiconet scheduling focuses on when a bridge is switched among various piconets and how a bridge node communicates with the masters in different piconets. This paper proposes an interpiconet scheduling algorithm named, hold mode based dynamic traffic priority load adaptive scheduling. The bridges are adaptively switched between the piconets according to various traffic loads. The main goal is to maximize the utilization of the bridge by reducing the bridge switch wastes, utilize intelligent decision making algorithm, resolve conflict between the masters, and allow negotiation for bridge utilization in HDPLIS using bridge failure-bridge repair procedure . The Hold mode - dynamic traffic - priority based - load adaptive scheduling reduces the number of bridge switch wastes and hence increases the efficiency of the bridge which results in increased performance of the system.

Keywords:

Bridge, Interpiconet Scheduling, Scatternet

1. INTRODUCTION

Bluetooth is a low-cost, low-power, and short-range radio technology used for wireless personal area networks. It operates in the unlicensed 2.4 GHz ISM band, frequency hopping spread spectrum (FHSS). The hopping frequencies cover 79 channels, each channel being 1 MHz wide. A piconet is a basic structure in bluetooth, which is constructed in an ad hoc fashion by one master and up to seven active slaves. A piconet can only contain one master and the master administers the whole piconet. A slave may connect to more than one master. A slave connecting to two or more masters is called a bridge. A set of piconets that are interconnected by bridges is referred as a scatternet. Although a bridge can participate in two or more piconets, it can only serve in one piconet at a time. The bridge will switch among all connected piconets in a time-sharing fashion. Several piconets can be interconnected via *bridge nodes* to create a *Scatternet.* Bridge nodes are capable of time-sharing between multiple piconets, receiving packets from one piconet and forwarding them to another .A bridge node can be a master in one piconet and act as slave in other piconets called *Master/Slave Bridge*. Alternatively, if a bridge node acts as a slave in all the piconets in which it is connected to, it is called *Slave/Slave* Bridge.

The scheduling of bridge switching among piconets is referred to as interpiconet scheduling. Obviously, an ill-considered scheduling may cause severe system degradation. An interpiconet scheduling algorithm can be developed and be well designed so as to help the bridge switch efficiently among piconets. On the other hand, the intrapiconet scheduling is referred to as the scheduling of a master serving the slaves connected by that master. Polling is a general scheme adopted for intrapiconet scheduling.

The main issue in inter-piconet scheduling is the switching of bridge node between piconets. Since each bluetooth device has one transceiver, it can participate only in one piconet at a time. As each master uses its own local clock, a bridge node has to re-synchronize with new master when it switches to a new piconet. The switch between two piconets may result in a *slot loss*. Another problem occurs when two masters try to access the bridge node simultaneously. This is referred to *as bridge node conflict.* Since a bridge node can listen to only one master at a time, the other master will not be able to communicate with the bridge node and will waste slots for polling operations.

In this paper we propose HDPLIS to eliminate bridge node conflicts and bridge switch wastes. In multi bridge scheme, a master adapts to increase traffic at a particular bridge i.e. by making a transfer to another bridge containing medium traffic via dynamic traffic procedure. This is done by calculating the overall traffic of the master. In the topology there are 3 piconets sharing a bridge, when two masters try to access the bridge node simultaneously, assuming a single bridge scenario, traffic value is calculated with respect to load adaption in every master. Then, the masters communicate their traffic value to the bridge, for predicting which master has maximum load as per the topology. If more than two masters have the same load, master conflict is said to occur. In a multi bridge scenario, much similar to single bridge scenario, the master conflict problem arises, but here each master maintain its' own dynamic traffic value. When this information is sent to the bridge, the bridge checks the dynamic table value from its piconets. If more than one master has the same dynamic traffic value, every such master attempts to access the bridge simultaneously. In this situation the master conflict is said to occur.

This paper eliminates master conflicts by using negotiation based bridge switch procedure. To avoid this master conflict problem in a single bridge scenario, we utilize the total traffic time (current load) and queue consume time of all masters. From this we calculate the bridge slot time for every master.

Total traffic value is associated with its estimate upcoming traffic value. If master 101 is greater than 102 means the bridge continue with 101. If not, bridge checks the traffic value from its piconets. If any one of them has max traffic (102), the bridge switches to 102 from 101. Thus, the bridge is switched among the piconets based on traffic value, as a measure to avoid master conflict problem.

In multi-bridge bluetooth scenario the master conflict resolution is calculated based on dynamic traffic values. Here, the problem lies in association with active masters!. If more than one active master has equal traffic then it is allowed to enter into the negotiation procedure. This is followed by bit vector

procedure. As per this procedure 0 is assigned to a bridge which is not willing to allow the other bridge's master due to high congestion. 1 is assigned to a bridge if it is willing to allow other bridge's master which is an indicative of low congestion. Negotiation procedure analyses which master has maximum dynamic threshold value via intelligent decision approach and informs the same to the bridge.

2. RELATED WORK

The interpiconet scheduling problem has been extensively addressed in the past, and several scheduling algorithms on this issue have been developed [1], [5], [8], [10], [11]. These algorithms schedule the presence, sequence and duration for which the bridge stays in every associated piconet and should coordinate with intrapiconet scheduling algorithms. In [13] Zhang and Cao proposed a 'Credit Based Scheduling (CBS)' scheme, which focuses on the fair link bandwidth allocation in node. Each node assigns credits to each of the connected link. The credits are allocated according to link utilization. Since each node makes its own decision to communicate with other node according to local share of the bandwidth, bridge node conflict may occur sometimes.

In [10] Vojislav Misic et al proposed a 'scatternet scheduling algorithm', which utilizes a pseudo random sequence to define the start time for all the meetings between two nodes. In addition, individual node will skip some of the meeting time based on traffic change. However since two nodes do not guarantee the meet at the same time, it may result as a miss of meeting time between two nodes.

In [14] Raymond Lee and Vincent Wong proposed the utilization of a 'flexible scatternet-wide scheduling' scheme, which places an adjustable switch table on each bridge node and master. Both bridge node and master decide when they can communicate with each other. Since this scheme gives a higher priority for the traffic on bridge nodes, it may not maintain fairness among all nodes.

In [12] Cordeiro et al proposed a 'locally coordinated scheduling algorithm' scheme which schedules the meeting time based on the traffic conditions. Each time before a node terminates the meeting with the connected node, they will negotiate the start time and duration time for the next meeting. This scheme does not consider the fairness among the nodes.

In [5] Har-Shai et al proposed a 'load adaptive algorithm', which utilizes decision variables to determine the period for a bridge node to stay in each piconet. Although the time for a bridge node to spend on a piconet can be adapted to traffic change, this scheme focuses on small-scale scatternets.

In [10] Racz et al proposed a 'distributed scatternet-scheduling algorithm', which allocates bandwidth to every link based on traffic estimation. Each time when a master meets with a bridge node, they will negotiate their next meeting time based on local traffic estimation. However in a dynamic environment, it is difficult to predict accurately the future traffic. Traffic estimation will affect the performance.

In [6] Jang-ping Sheu et al proposed a 'traffic aware scheduling' for bluetooth scatternets, in which serving master is responsible for making a decision when to switch the bridge. Bridge can be switched between the masters effectively based on the traffic conditions. But this is applicable for only small-scale scatternets where a single bridge is shared by multiple piconets.

In [1] Baatz et al proposed a Priority based Inter piconet scheduling algorithm for bluetooth scatternets, which maintains a priority queue at the bridge node for taking decisions to switch the bridge among various piconets. The Priority queue maintains the priorities of piconet masters based on current traffic conditions. Bridge is intelligently switched among various piconets with respect to traffic loads and reduces the packet transmission delays.

In [5] Har-Shai et al proposed a 'Load adaptive Inter piconet scheduling algorithm' which utilizes the hold mode, and its implementation does not require modifications to the Bluetooth specifications. It manages the scheduling mechanism of the bridge. It determines the duration of bridge activity in the different piconets such that the delay incurred by packets requiring inter-piconet routing is reduced. The algorithm adapts to varying values of load by using information regarding the bridge's queues to different masters.

In [9] Lin and Tseng proposed an 'Adaptive Interpiconet Scheduling Algorithm' Based on Sniff Mode in Bluetooth Scatternets to reduce the average interpiconet packet delay while increasing the utilization of a bridge. This scheme estimates the time duration for which the bridge should stay in each piconet according to the traffic pattern so that the bridge can avoid being idled as possible.

In [4] Ching-Fang Hsu, and Shu-Ming Hsu proposed an adaptive interpiconet scheduling algorithm based on HOLD mode in bluetooth scatternets. Motivated by the above literature, here we propose a new interpiconet scheduling algorithm based on the HOLD mode—a power-saving mode of Bluetooth—which includes dynamic load priorities to reduce the average interpiconet packet delay while increasing the utilization of a bridge. This scheme estimates the time duration for which the bridge should stay in each piconet according to the traffic pattern so that the bridge can avoid being idle as possible.

3. MOTIVATION

Existing work records poor bridge usage and performance due to transmission delays. A bridge wastes lot of memory for calculating static threshold. The goal of this paper is to dynamically switch the bridge according to its master's traffic conditions thereby reducing bridge switch wastes. In [5], Har-Shai et al. proposed a scheduling algorithm based on the HOLD mode, another power-saving mode in Bluetooth. Nevertheless, this innovative algorithm only works in a two-piconet scatternet. The HOLD mode is the choice of a power-saving mode that can be employed to implement interpiconet scheduling algorithm. Comparing the SNIFF mode with the HOLD mode [9], the major difference is that the hold period is negotiated by 'a master and the bridge' each time the bridge enters the HOLD mode, whereas the sniff interval of the SNIFF mode is set only once and does not change for a long time [5]. Consequently, as applied to interpiconet scheduling, HOLD mode based algorithms provide more flexibility than SNIFF mode-based algorithms, although it comes with the price of one extra slot for the negotiation of hold period.

Moreover, the most critical issue that HOLD-mode-based interpiconet algorithms have to deal with is how to accurately predict the hold period for an individual associated piconet. In our proposed work we consider both large and small scale scatternets. For study, we have taken 2 bridges with 6 piconets, a sort of multi bridge scenario.

According to [6] serving master is responsible for making a decision when to switch the bridge. Bridge can be switched between the masters effectively based on the static traffic conditions. But this is applicable for only small-scale scatternets where a single bridge is shared by multiple piconets but our proposed scheme achieves multi bridge scenario where a single bridge might be shared by multiple piconets.

In [5] the serving master manages the scheduling mechanism of the bridge. It determines the duration of bridge activity in the different piconets such that the delay incurred by packets requiring inter-piconet routing is reduced. The algorithm adapts to varying values of load by using information regarding the bridge's queues to different masters. In a nutshell, the ideas were a combination of priority based scheduling and load adaptive inter piconet scheduling algorithm.

In the proposed work, the bridge is adaptively switched between the piconets with various traffic loads. The main goal is to maximize the utilization of the bridge by reducing the bridge switch wastes. We make use of intelligent decision making algorithm for resolving conflict between the masters, and providing for negotiation for bridge utilization in multi bridge scenario. Negotiation between the bridges is done to implement bridge switch procedure between the bridges and reduces the master conflict problem in multi bridge scenario. The Hold mode based dynamic priority load adaptive scheduling reduces the number of bridge switch wastes and hence increases the efficiency of the bridge, and thereby the performance of the system.

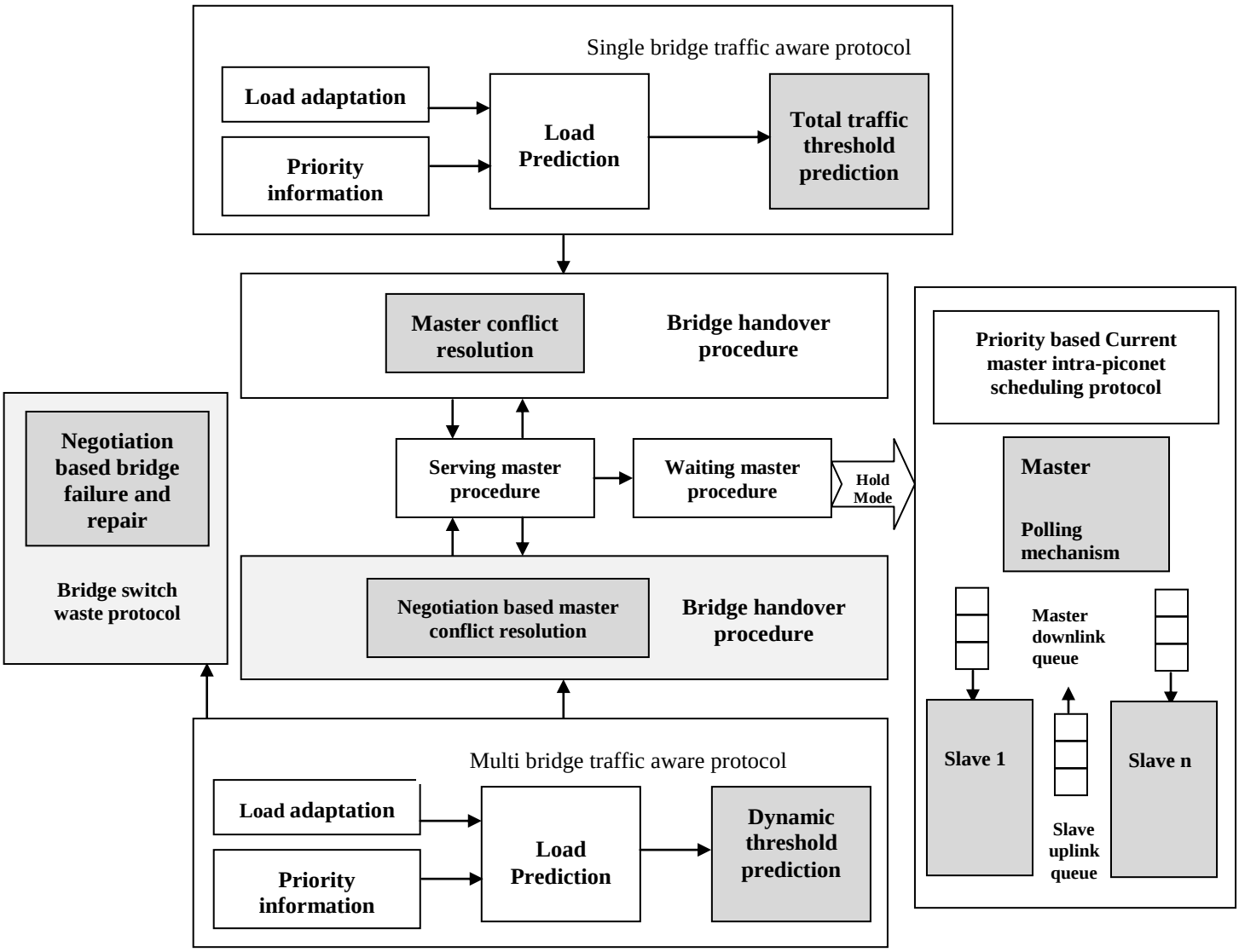

Fig.1. HDPLIS Architecture

4. HDPLIS ARCHITECTURE

The key modules of the HDPLIS system are:- serving master process, bridge handover process (normal & negotiation based master conflict resolution), waiting master process, single and multi bridge traffic aware protocol based on intelligent decision making algorithm, Negotiation based bridge switch process in multi bridge scenario. All the modules take the traffic threshold information and the scheduling table from their repositories. When we integrate all these modules overall packet transmission delays and bridge switch wastes are reduced thus increasing the efficiency of the bridge and hence the throughput.

4.1 HDPLIS SYSTEM DESIGN

Under the following conditions, serving master i has to release the usage of the bridge to the waiting master j.

(C1): $WT_j > WT_{th}$, (TIME event),

(C2): $(QCT_j + \alpha_j * WT_j) > (QCT_i + Qcthold)$ (QUEUE event).

(C1) implies that master j has been waiting for the bridge past the WT_{th}. (C2) implies that all the data required to be transmitted completely between master j and the bridge is larger than those between masters i and the bridge plus Qcthold. The Qcthold is designed for avoiding the ping-pong effect when QCT_i and QCTj are too close to each other. (C1) is used to avoid excessive transmission delay of the waiting master. The released event triggered by this condition is termed as TIME event. (C2) is used to allocate more service time to the link with high traffic loads. The released event triggered by this condition is termed as QUEUE event.

If none of the two conditions are satisfied, then the serving master i can keep using the bridge. This is termed an EXTEND event. It is worth mentioning that an EXTEND event will result in a failed unhold for the waiting master which has the highest possibility of getting the usage of the bridge in the near future (here, this implies waiting master j).

However, the EXTEND event implies that the traffic load for waiting master j is not larger than the load of serving master i by a prespecified threshold. To improve the throughput of a scatternet, the master with high traffic loads will be allocated more service time. However, when an EXTEND event is triggered, it also implies that the LT_j of the waiting master j expires. Therefore, master j will try to unhold the bridge on the hold slots in the future. Consequently, the LT_j in the scheduling table of the serving master i must reset to Thold. The bridge receiving the scheduling table and dynamic table and being informed by the serving master i to switch to another piconet to serve the new serving master j.

If any one of the bridge handles over traffic, the master is handed over to the next bridge using dynamic table information. In this hand over procedure, we have 2 types of solution: Single bridge master conflict resolution and multi bridge negotiation based master conflict resolution. In this paper an interpiconet scheduling algorithm based on the power saving mode HOLD without any modifications to the bluetooth specification has been proposed. Our approach addresses the reduction of packet delay time and improvement on the utilization of a bridge.

Negotiation based bridge switching procedure (in multi-bridge) suggests which bridge will have to serve first using dynamic traffic threshold value. If any one of the serving bridges is under traffic over flow (like failure), the serving bridge's active masters switch to other bridges under negotiation procedure. In multi bridge scenario, scatternets resolve master conflict problems by following the *bridge hand over* procedure.

The serving and waiting master procedures are used when the serving master decides to release the usage of the bridge; it has to update its traffic information in the scheduling table. The bridge will transfer the scheduling table from the old serving master to the new serving master. According to the scheduling table, the new serving master can figure out the time it can use the bridge, and the waiting master can calculate the time it need not to poll the bridge in the following hold slots. In this paper we predict the total and dynamic traffic value from current load of the master, estimate the upcoming load, and also analyses the historical traffic information of each master.

5. HDPLIS TABLES

HDPLIS is operated on a constructed scatternet. Only ACL (Asynchronous Connectionless) link is considered for the connection between a master and a slave. In HDPLIS, the hold mode is used as the operating mode for the bridge to switch among piconets. The hold interval negotiated by a bridge with its serving master is Thold. In HDPLIS, each master maintains a scheduling table and dynamic traffic table.

5.1 SCHEDULING TABLE

Scheduling table contains the traffic information of all masters that the bridge is connected to. When the serving master decides to release the usage of the bridge, it has to update its traffic information in the scheduling table. The bridge will transfer the scheduling table from the old serving master to the new serving master. According to the scheduling table, the new serving master can figure out the time it can use the bridge, and the waiting master can calculate the time it should avoid polling the bridge in the following hold slots. Therefore, with the scheduling table, each master can record its traffic information in the table and obtain the traffic information of the neighboring masters at the same time. The scheduling table is very helpful in designing the HDPLIS scheme.

A scheduling table is shown in Table 1, where MID represents the identity of the master and LT_i, QCT_i, WT_i, and α_i are the traffic information of master i. The details of the fields in the scheduling table are described below. The scheduling table includes the following fields: MID**:** the identity of the master, QCT: (Queue Consuming Time): the estimated time that a link will need the bridge to serve, LT: (Lost Time): the estimated time that a master cannot get the usage of the bridge, WT: (Waiting Time): the time that a master has been waiting for the usage of the bridge. α : the historical information of, on average, the traffic generation rate per slot between the master and the bridge.

Table.1. Scheduling Table (Initial state)

MID	QCT	WT	LT	α
1100	22.0	0.0	22.0	0.2
1101	88.0	12.0	0.0	11.0
1102	44.0	9.0	4.0	00.5
1200	55.0	11.0	12.0	00.5
1201	55.0	10.0	0.0	00.8
1202	22.0	3.0	4.0	00.8

QCT is defined as the time that a link needs to transmit all the data packets in the queues of the master and the bridge. There is a queue agent to monitor the status of the queue on either side of a link. The bridge will notify the master about this information at each communication with the master. Based on this information, the master can obtain the QCT.

5.1.1 Scheduling Table Parameters:

LT is defined as the time that a master cannot use the bridge. The QCTs of all masters connected by the bridge are stored in the scheduling table. When the serving master has to release the usage of the bridge, according to the QCTs, the serving master can predict the duration from the time it releases the bridge to the time it obtains the bridge next time. This duration is called LT. LT can be used to reduce the number of failed unholds of the waiting masters. For example, when master' A' has to release the usage of the bridge to master 'B', master 'A' will compute the LT_A to predict how many time slots that it may lose the usage of the bridge in the future. Thus, after master 'A' releases the usage of the bridge, master 'A' will skip the hold slots during the LT_A. Therefore, master 'A' can reduce the number of failed unholds.

WT is the time that a master has been waiting to acquire the usage of the bridge since its previous release. 'A' represents the history of traffic loads, which is defined as the historical information, on the average, i.e. traffic generation rate per slot between the master and the bridge. Since the decision of the master to release the bridge depends mainly on the value of QCT, the precision of QCT will influence the performance of HDPLIS. Therefore, to obtain a precise QCT, the history of traffic loads is counted so as to evaluate the QCT due to the temporal locality of the traffic. Let α be the increment of the traffic in queue during a fixed time period, say T. The queue agent responds to maintain q. Thus, α can be obtained as q/T. After α is obtained, the queue agent will reset q to zero.

When the serving master has to release the usage of the bridge, it records α in the scheduling table. Hence, when the new serving master gets the usage of the bridge, it can evaluate the QCT more precisely for a waiting master. We have introduced how to obtain QCT precisely by means of α. In the ensuing paragraphs, we will explain how to obtain LT by means of QCT and α. LT refers to the time that the serving master will not get the bridge after it releases the usage of the bridge. When the serving master i has to release the usage of the bridge, it will find a candidate to be the new serving master, say 'j', and will update the LT_i. The serving master i first find the minimum LT_j from the scheduling table for some 'j'.

If there exists more than one minimum LT, then it selects the one with the maximum WT. This means that the waiting master 'j' has the highest priority to get the usage of the bridge once the serving master releases the bridge. The serving master has to update LT_i once it decides to release the usage of the bridge to the new serving master 'j'. However, QCT_j in the scheduling table of master 'i 'is an outdated value since it was recorded when the master 'j' has released the usage of the bridge. Therefore, it does not stand for the current traffic loads of master 'j'. As a result, we can use α_j to roughly estimate QCT_j. Therefore, the time that the serving master i will not get the usage of the bridge, let's call it LT, can be obtained as follows:

$$LT = QCT_j + \alpha_j * WT_j \quad (1)$$

If a serving master 'i' gets the usage of the bridge, it first finds the minimum LT_j from the scheduling table, for some 'j'. According to this information, master 'i' will know how much time it has been allowed to use the bridge freely. In addition, master 'i' is responsible for the maintenance of the scheduling table. That is to say, serving master 'i' should add 1 to each WT and subtract 1 from each LT, per slot, in the scheduling table.

When $LT_j = 0$, master 'i' must check if it has to release the bridge to the waiting master 'j'. When the release condition is satisfied, the serving master i has to release the usage of the bridge to the waiting master 'j'. Serving master i then perform the serving master part of the bridge release procedure. As described above, once serving master i intends to release the bridge, it will calculate LT_i by means of the scheduling table. After the LT_i is calculated, master 'i' updates LT_i in the scheduling table and resets the WT_i to zero. Master 'i' then transmits the scheduling table to the bridge, and informs the bridge to serve the new serving master 'j'. The role of master 'i' is turned from being a serving master to that of a waiting master. Therefore, afterwards, master 'i' will perform the bridgeless phase. The bridge receiving the scheduling table will perform the bridge part of the bridge release procedure as well. The bridge then waits for being unholded by the new serving master 'j' and maintains the scheduling table during this waiting period.

Maintenance period means that the bridge will record the time slot count (sc) during the period from the time it returns an ACK to the old serving master to the time it returns another ACK to the new serving master, acknowledging the unhold of the new serving master. The period should include the guard time difference between the old and the new serving masters. When the bridge is unholded by the new serving master, it subtracts slot count from each LT, adds sc to each WT in the scheduling table, and then transmits the scheduling table to the new serving master.

5.2 DYNAMIC TABLE

Dynamic table contains each master's total traffic, along with that of dynamic traffic. The dynamic table is shown in Table.2 which includes MID (master identity number for all masters in all bridges), total_traffic (all masters total traffic (calculating from estimated traffic and historical traffic), dynamic traffic (including all masters for using negotiation procedure).

Table.2. Dynamic table (Initial state)

MID	Total_Traffic	Dynamic_Traffic
100	8.6	2.8
101	2.2	3.6
102	5.2	5.3
200	5.2	1.7
201	4.8	3.3
202	7.8	5.9

5.2.1 Dynamic Table Parameters:

Total_ Traffic is defined by calculating the masters' traffic from its history of traffic; estimating the upcoming traffic, and also taking into account the current traffic load. The purpose behind calculating Total_ Traffic is to find the dynamic traffic.

Master's total traffic is calculated as,

$$\text{Total\_Traffic (TM)} = \phi M + ET \quad (2)$$

where,

ΦM is historical traffic of master

ET is estimated traffic of master's upcoming traffic

Dynamic traffic threshold is defined by calculating the masters' dynamic traffic load based on its total traffic. In this method, the traffic calculated is used for negation based bridge switch procedure which is performed to solve master conflict problem in a multi-bridge scenario.

$$DTT = \text{avg}\,[PT\{(CLi + \phi M) + \sum(CLj + \phi M)\}] \quad (3)$$

where,

DT (dynamic traffic table) contains all masters traffic info

CLi is Current load of master

ϕM is history traffic info of master

$$\text{Estimated traffic is calculated as } ET = 2 * \alpha \quad (4)$$

α is the historical information of the traffic generation rate per slot between the master and the bridge taken on average.

Algorithm 1: Serving Master Procedure

This procedure is proposed by [6] TASS

{The serving master should execute the algorithm *per slot*}

Step 1: The serving master, say *i*, maintains the scheduling table. The task performed is to add 1 to every *WT*, subtract1 from every *LT* (for all waiting masters), and update the QCT_i in the scheduling table according to its queue status.

Step 2: If there is no data to send between the serving master *i* and the bridge then

Execute the *Bridge Release Procedure*.

End if

Step 3: If no other *LT* except LT_i in the scheduling table is equal to zero then

Go to Step 8.

End if

Step 4: Choose a waiting master *j* with $LT_j = 0$.

If there are more than one waiting master with *LT* = 0 then

Select the waiting master *j* with the largest *WT* and the other *LT*s are reset to *Thold*

End if

Step 5: if $WT_j > WT_{th}$ then

Execute the *Bridge Release Procedure*

Go to Step 8.

End if

Step 6: if $((QCT_j + \alpha_j * WT_j) > (QCT_i + Qcthold))$ then

Execute the *Bridge Release Procedure*

Go to Step 8.

End if

Step 7: Reset LT_j to *Thold* Go to Step 8.

Step 8: End.

Algorithm 2: Serving Master Bridge Release Procedure

This procedure is proposed by [6] TASS

The part to be executed by the serving master.

{The serving master *i* deciding to release the usage of the bridge will perform the following operations}

Step 1: Calculate LT_i.

Step 2: Update LT_i and reset WT_i to zero in the scheduling table.

Step 3: Transfer the scheduling table to the bridge and inform the bridge to be unholded by the new serving master.

Step 4: Wait for the ACK from the bridge.

Go to the *waiting master procedure*

The bridge executes the following procedure.

{The bridge receiving the scheduling table and dynamic table and being informed by the serving master i to switch to another piconet to serve the new serving master j .if any one of the bridge goes to over traffic , the master hand over the next bridge using dynamic table information will perform the following operations}.

Algorithm 3: Scheduling Table Handover Procedure

In this hand over procedure, we have 2 types of solution. They are,

- *Single bridge, master conflict resolution algorithm*
- *Negotiation based multi bridge master conflict resolution algorithm*

T_m = master's traffic

DTT_m = dynamic traffic threshold of all masters

ST = scheduling table of master

QCT = queue consuming time of total bridge utilization

st = slot time of every master

E_T = estimated traffic

B_0 = active bridge

B_i = other bridges

B_D = common master

Single bridge:

Predicting all masters($T_{m)}$ traffic from ST calculate QCT of all masters

Calculate the st of every master (technique used)

Predicting the $E_T = 2 * \delta$

If (Ti is greater than Tj) then

the bridge continue the active master

Else switch to master with high traffic

Else if the active master is less than some other master, then the bridge switch to max traffic of master

Else if (active master is less than some other master but the other masters has same traffic)

The bridge will decide to alter the priority based on the concept of ageing

The master with the highest priority will serve the bridge

Else the bridge goes to FCFS procedure

End if

Multi bridge:

For (B=1; B<=n; B++)

Calculate DTT

If check B_i [DTT] < B_0

Add B_i to bit vector procedure

Else

Skip the bridge

Else if

Some B_i [DTT] < B_0 and B_i [DTT] are same

add B_i to bit vector procedure

Goto bit vector procedure

{

In bit vector procedure

0→ represent bridge is not willing to allow the B_0's masters

1→ represent bridge is willing to allow to B_0's masters

}

Assign 0 has max congestion

Assign 1 has min congestion

For (i=1; i<=n; i++)

If (B[i]=0) then

Goto next bridge until B[i]=1

Check B[i]=1 then

Assign B[i] = B_0

else if

All B[i] = 0 then

Goto negotiation procedure

else if

Some B[i] equal with 1's then

Goto negotiation procedure

Negotiation procedure:

All B[i] assign to 0 or assign 1

B_D has all priority information of all bridges

Assign i= no of bridges

While (i)

{

Delay (1000) // millisecond

For each bridge in the network

Store the details of that bridge in a temporary variable

}

Store the temp value into B_D

B_0 = send request to B_D

On receiving the request from B_0 , B_D will send the priority information to B_0

If (any one bridge has max priority on comparison with others) then

B_0 switch to highest priority of bridge

From the scheduling table, traffic from all masters is estimated. The parameters used to do the same are LT, WT, QCT, and Traffic coefficient. Following this, the calculation of QCT is performed. Every master then will calculate their slot time following which a value for estimated traffic is generated. This value of estimated traffic is used to find the total traffic.

Algorithm 4: Waiting Master Algorithm

This procedure is proposed by [6] TASS

{

The waiting master should execute the algorithm per slot. Suppose the waiting master is master j, for some j.

}

if LTj > 0 then

LTj = LTj − 1

else

Back to the normal operation of hold mode.

{

It implies that the master j will try to unhold the bridge on the following hold slots.

}

End if

If master j unholds the bridge successfully then

Go to the serving master procedure

End if

Algorithm 5: Negotiation Based Multi Bridge Master Conflict Resolution

B_i [T] = traffic status of bridge

Assign i = number of bridges

B_D = common master

B_c = activate bridge

The traffic information of all bridges is stored in B_D

While(i)

{

Delay (1000) // millisecond

For each bridge in the network

Store the details of that bridge in a temporary variable

Assign B_c = temp

}

Store all the B_c's values

Check all bridges' traffic value and identify the bridge with maximum traffic

Assign the max traffic value of bridge into B_c

If (B_i [T] has max traffic, i is identified as having more traffic on comparison with the others) then

$B_c = B_i$ [T]

Else if (some B_i [T] has same traffic) then

Goto B_D table

Assign number of jobs per slot = st

{

For all i calculate traffic value based on QCT & st

Update priority of bridge per delay (1000)

Predict priority of each bridge before & after the time intervals

Store the priority value of each bridge in B_D

}

Predict max priority of bridge (aging technique) from B_D

Assign max priority of bridge into B_c

Else

Check total number of masters served by each bridge

Assign bridge with maximum number of masters to B_c

End if

6. RESULTS

In multi-bridge bluetooth scatternet simulation scenario multiple bridges exist with their piconets. If any one of the bridge fails, the failed bridge's working masters have to transfer their packets via other bridges optionally. As a case study, our multi bridge bluetooth scenario has 2 bridges with 6 piconets each has 3 piconets individually. In this piconet topology master handles average load. QCT values are assumed normally.

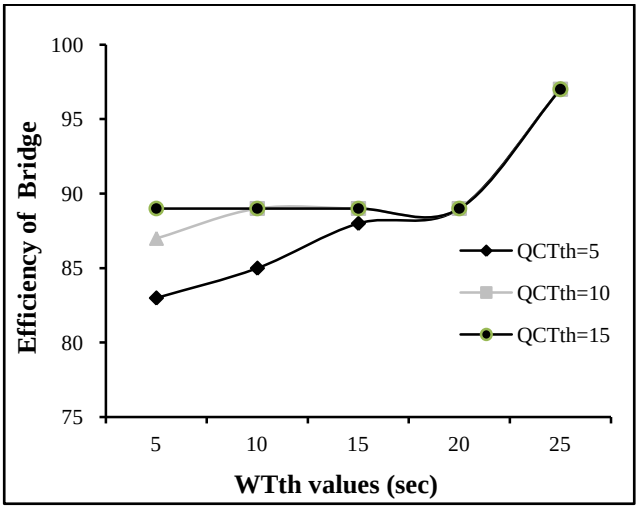

Fig.2. Efficiency of the bridge Vs WT_{th}

Fig.2 shows the efficiency of the bridge at QCThthold=5, QCThthold=10, QCThthold=15. Since the total activity ratio will increase with the increase of WT_{th}, the efficiency of the bridge also increases with increase in WT_{th}. Similarly, the throughput for the low QCThthold value is worse than those for the high QCThthold values. However, if the QCThthold is too large, it will cause the serving master with a low traffic load not to release the bridge. Therefore, the best performance is recorded when QCThthold is 10 and when the WT_{th} is large enough. From the above experimental results, we find that when WT_{th} and QCThthold are large enough (in the above experiment, $WT_{th} > 40$ and QCThthold > 5), the results are very close to each other and varying both parameters would not affect the performance significantly.

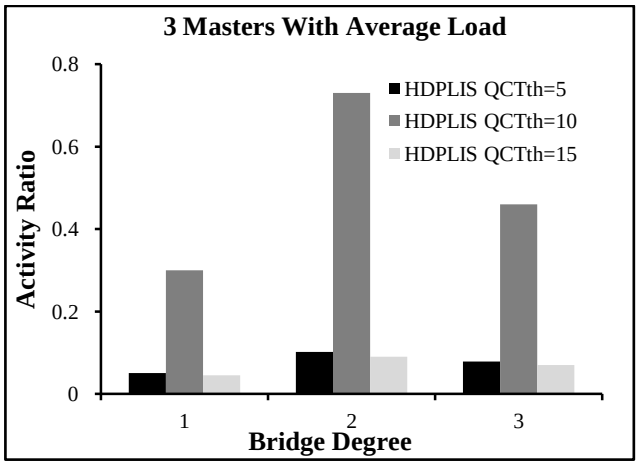

Fig.3. HDPLIS: Average Load interpiconet behavior

Fig.3 considers QCT values with average and peak load. Master with average load QCTth=10 has maximum activity ratio compared to that with 5 and 15. For 3 masters with average load and queue consume time value at 10, the activity ratio is increased. For master level 2 HDPLIS is performed well when compared with TASS. For 3 masters with peak load and queue consume time value 15, the activity ratio is increased at master level 2. Peak load activity ratio is increased while increasing QCTth value. Master level 2 has high bridge activity compared to that of 3 masters. So HDPLIS perform well compared to TASS.

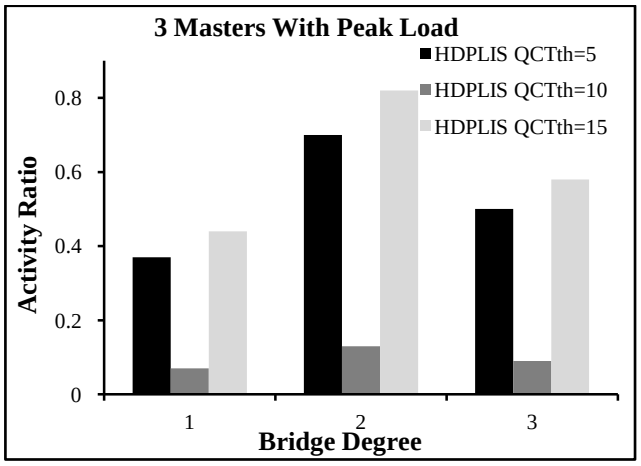

Fig.4. HDPLIS: Peak load interpiconet behavior

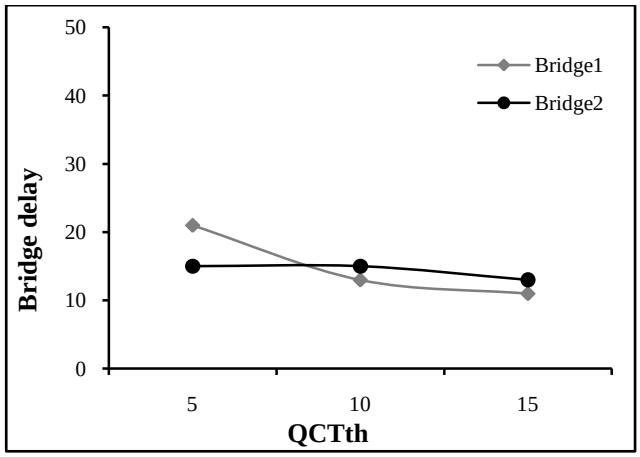

Fig.5. Bridge delay behavior

In Fig.5 by increasing QCT values with average load the bridge delay for bridge 1 is decreased with increasing consuming time. But for bridge 2 there is no immediate reflection in bridge delay. It has to change slowly to increase the consuming time. Bridge delay for bridge 1 and bridge 2 are very closer for both QCTth values assigned as 10, 15. but the behavior for QCTth value 5 is very different because by increasing the queue time the traffic load is decreased. In peak load, bridge 2 did not changed with increased QCT values. But the delay of bridge 1 is decreased with increasing QCT values.

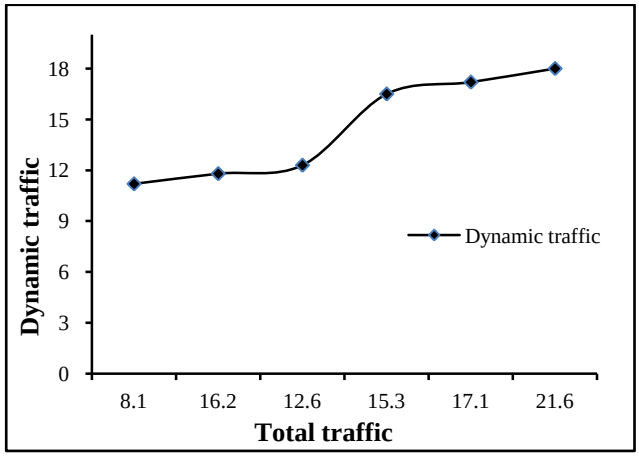

Fig.6. total traffic Vs dynamic traffic with consume time 5 sec

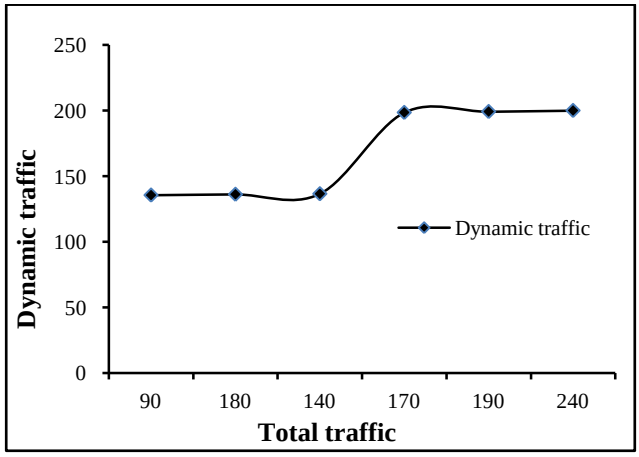

Fig.7a. Total traffic Vs dynamic traffic with consume time 15sec

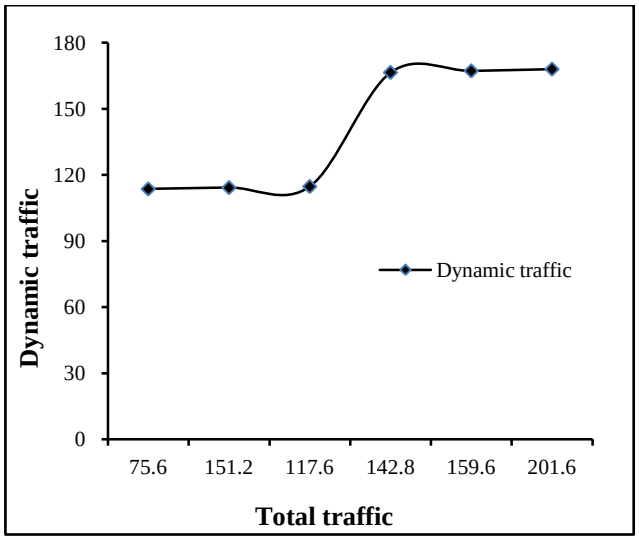

Fig.7b. Total traffic Vs dynamic traffic with consume time 10sec

In Fig.6, 7a, 7b, Master dynamic traffic depends with total traffic values; i.e. it continuously changes with queue time. The bridge transfers between the master using scheduling table parameters. If bridge is working with current master 101, the current master transfers its data to the bridge. The bridge receives the data, at last current master exceeds its serving time, and the master sends bridge release procedure to the bridge. The bridge receives the message and sends the polling message to other master. The bridge checks which master has the minimum lost time and allows that master to serve it at that instant. If the master 102 has minimum lost time, this implies that the bridge will now proceed to work with 102. The same procedure works in bridge 2. Master calculates its total traffic values from its upcoming traffic and history of masters. From the total traffic we have to estimate the dynamic traffic (average of all masters within the bridges. Here bridge1, bridge 2 are available with each 3 masters). The master displays its dynamic and total traffic to send to the bridge1. Bridge1 identifies masters' traffic values, if any one of the bridges are dropped due to heavy traffic the bridge1 identifies which bridge is available to transfers its master information through the network. Here bridge 2 is available to receive the bridge 1 traffic.

HDPLIS illustrates the effects of total traffic and QCTthold on the delay of the bridge. Since the total activity ratio will increase with the increase of QCTthold, the delay of the bridge also decreases with increase in QCTthold. Similarly, the throughput for the low QCTthold value is worse than those for the high QCTthold values. However, if the QCTthold is too large, it will cause the serving master with a low traffic load not to release the bridge. Therefore, the best performance is recorded at QCTthold = 10. In this situation the total traffic was found to slowly increase when compared to when the QCTthold value was 5. If the total traffic is reduced the delay of the bridge is also reduced.

The comparisons between HDPLIS and TASS (Traffic Aware Scatternet Scheduling [6]) on throughput, activity ratio are presented as well (Fig.8a and Fig.8b). The packet generation rates of the masters follow a constant bit rate (CBR). Among these masters, the packet generation rate of one master is fixed on 300kbps and those for the others are fixed to 60kbps. A high

packet generation rate implies that the master would need more bridge service time. The bridge does not generate any packets at all and the destinations of all packets are to the bridge. The simulation time is 100 seconds.

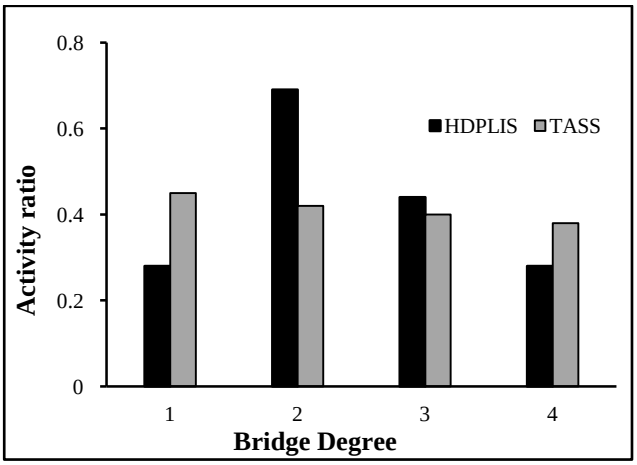

Fig.8a. Comparison of behaviors between 4 masters with average load

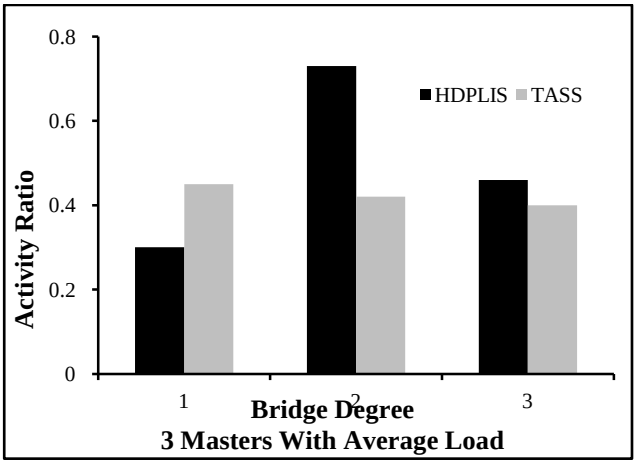

Fig.8b. Comparison of behaviors between 3 masters with average load

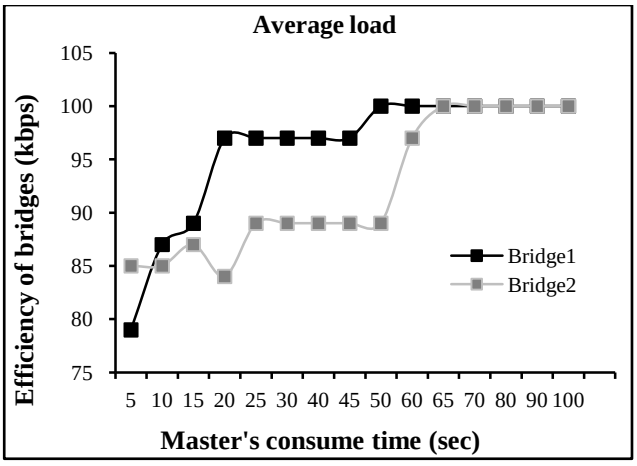

Fig.9a. The impact of the various traffic loads on the total throughput when a bridge connects to three masters average load

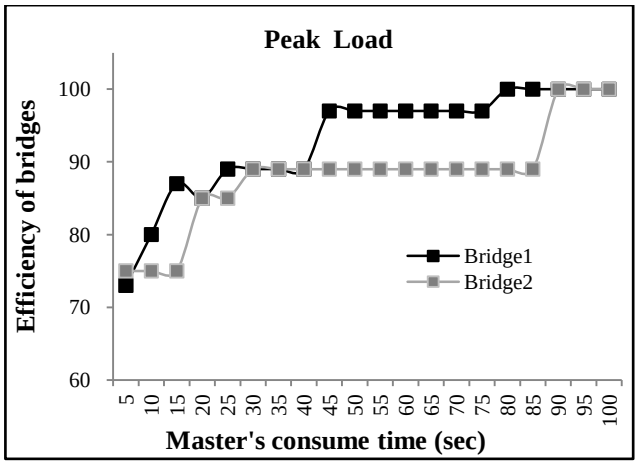

Fig.9b. The impact of the various traffic loads on the total throughput when a bridge connects to three masters maximum load

Fig.8a and 8b show the impact of the degree of the bridge on the activity ratio and the throughput (Fig.9) of the master whose packet generation rate equals 300kbps, respectively. The activity ratio means the ratio of the total bridge service time of the master whose packet generation rate equals 300kbps to the total simulation time. The throughput is evaluated by the data packets received by the bridge per second. Obviously, HDPLIS can allocate more bridge service time to the master with high traffic loads. The master with high traffic loads can almost obtain the maximum throughput. On the contrary, in TASS, the bridge service time allocated to the master with high traffic loads decreases seriously as the degree of the bridge increases. Accordingly, the throughput of the master with high traffic loads will decrease when the degree of the bridge increases as well. It is because that, in TASS, the bridge service time allocated to the masters is based on the link level fairness. That is, the chances of the masters getting the usage of the bridge are the same, no matter how heavy the traffic load of the master is. Therefore, the bridge service time of the master with high traffic loads will decrease seriously as the bridge degree increases. Contrarily, in HDPLIS, the master with high traffic loads will have higher probability to obtain the usage of the bridge due to QUEUE event.

On the other hand, HDPLIS will not cause the master with low traffic load to starve since the master with low traffic load can obtain the usage of the bridge by TIME event. In 4 masters with average load the activity ratio is highly increased with 2 masters. At the same time slowly it will decrease while adding new masters, but TASS slightly decreases adding new masters. For the case Bridge with 4 masters, the HDPLIS fails to compare with TASS. So HDPLIS is well suitable for 2 piconet only. Otherwise, activity behavior is same in both 3 and 4 masters with average load.

Fig.9 illustrates the total throughputs of HDPLIS and TASS, which are obtained from every 1600 slots (i.e., 1sec). As shown in Fig.9a HDPLIS and TASS can reach the maximum throughput in the first 20 seconds since the packet generation rates of the three masters are the same. In the following 20 seconds, the packet generation rate of one master rises to 400kbps. Since TASS does not take traffic information into

consideration, it cannot adjust the switch scheduling according to different traffic loads of masters.

Fig.9b Thus, HDPLIS can still keep the maximum total throughput, but TASS cannot. At the last 20 seconds, the packet generation rate of one master is reduced to 20kbps. As the figure shows, HDPLIS can adapt to the real traffic rapidly, but TASS still needs some time to adapt to the real traffic loads. Since there are still a lot of data packets queued at the previous 20 seconds in TASS, hence, it needs additional time to consume the queued packets. Therefore, the adaptability of HDPLIS is superior to that of TASS.

Bridge 1 and bridge 2 comes with average load with 3 masters. Bridge 1 and bridge 2 in their initial stage are very closer with equal efficiency but when time increases, bridge 1 has high efficiency compared to bridge 2. Because bridge 1 has medium traffic, by increasing service time the efficiency of bridge also increases, at the end of serving time bridge 1 and bridge 2 are again it will closer to each other. Bridge 1 and bridge 2 comes with peak load with 3 masters. Initially, both are very closer with equal efficiency but with increase in time the bridge 1 has high efficiency compare with bridge 2 because bridge 1 has medium traffic so increasing service time the efficiency of bridge also increases at the end of serving time bridge 1 and bridge 2 are again it will closer to each other.

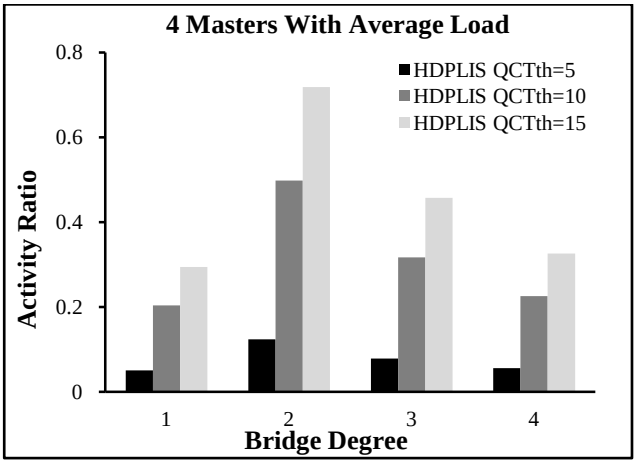

Fig.10a. Comparison of interpiconet behavior: 4 masters with average load

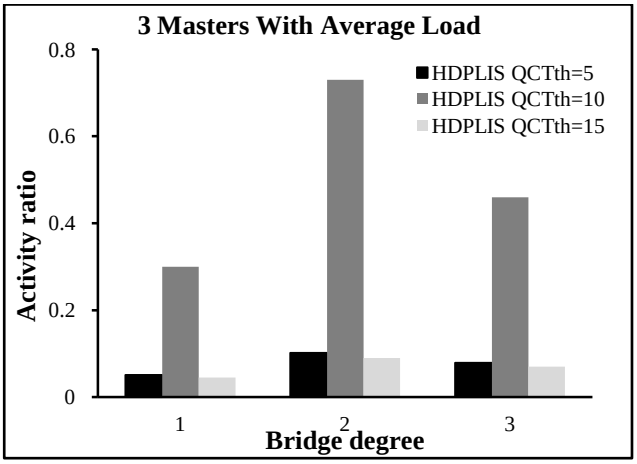

Fig.10b. Comparison of interpiconet behavior: 3 masters with average load

In Fig.10a and 10b, in the case of 4 masters with average load, the activity ratio is increased while increasing QCTth values, resulting in high performance. Activity ratio attains its maximum only when QCTth=10. Changing QCTth values does not increase activity ratio. Rather this gives poor performance. In Fig.11a and 11b the bridge efficiency criteria for 4 masters with average load is better than the case of 3 masters because increasing the queue time the traffic is reduced heavily. Bridge 1 and bridge 2 curves are very closer. Bridge efficiency is increased only at maximum consuming time of queue. Curves are not closer to each other. Bridge 1 and 2 has maximum bridge utilization compare than 4 masters.

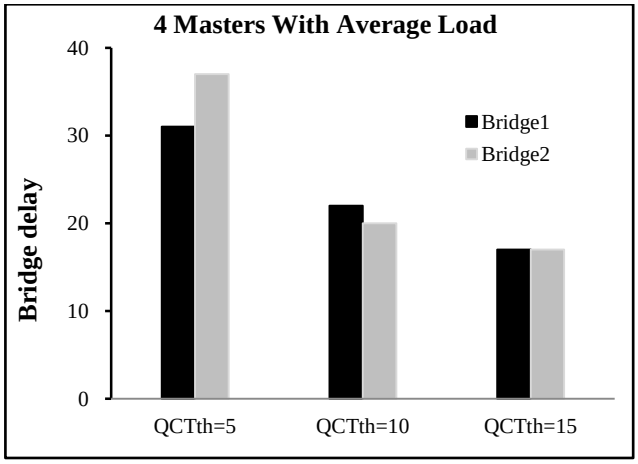

Fig.11a. Comparison of efficiency: 4 masters with average load

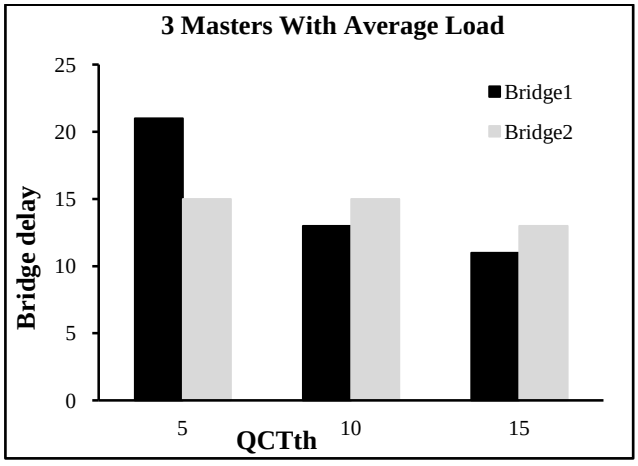

Fig.11b. Comparison of efficiency: 3 masters with average load

7. CONCLUSION

This paper presents a hold mode based dynamic priority load adaptive Interpiconet Scheduling (HDPLIS) scheme, which can dynamically adjust the bridge service time according to a master's traffic load, reduce the number of failed unholds time, and further increase the system's throughput.

The primary idea of HDPLIS lies in estimating the dynamic traffic to solve the problem of negotiation. That is, HDPLIS allocates enough bridge service time to the master with a high traffic load and reduces the bridge switch wastes. At the same time, to avoid excessive transmission delay of the master with a low traffic load, HDPLIS will allocate the bridge service time to

a master once that master has waited for a period of time, but no longer than WT_{th}. Dynamic traffic will reduce the master service time if the current bridge fails.

Each master generates traffic values to transfer the bridge. Bridge will need to handle negotiation process for its master conflicts problem. (very large and very small values of WT_{th} result in poor efficiency of the bridge. In order to achieve optimal efficiency, it is desired to choose an intermediate value so as to keep the waiting period of the master normal). This hold period is negotiated by a master and bridge, each time the bridge enters the hold time for every slot period.

Once the slot period is encountered the bridge will change its hold period each time. So finally we have to apply QCTthold. WT_{th} is kept at normal (no max and min values), to achieve good efficiency of bridge throughput and reduced packet transmission delay. In this paper, HDPLIS has been shown to perform well for two bridges shared by multiple piconets. However, it is possible that more masters share more than one bridge (bit vector procedure for multi bridge scenario). Therefore, for the sake of completeness, a comprehensive investigation should be made in the future to find out how HDPLIS performs when multiple bridges are shared by multiple piconets, at the same time enabling masters (waiting masters) to remain in Hold mode to do simultaneous Intra Piconet scheduling proposed for efficient bridges in bluetooth scatternet, as well as a negotiation based bridge-switch process between multiple bridges.

REFERENCES

[1] Baatz. S, Frank. M, Kuhl. C, Martini. P and Scholz. C, "Bluetooth scatternets: An enhanced adaptive scheduling scheme,**",** in *Proceedings of the IEEE INFOCOM, the Annual Joint Conference of the IEEE Computer and Communications Societies*, Vol. 2, pp. 782–790, 2002.

[2] Capone. A, Gerla. M and Kapoor. R, "Efficient polling schemes Bluetooth Picocells", *Proceedings of the IEEE INFOCOM, the Annual Joint Conference of the IEEE Computer and Communications Societies,* Vol. 7, pp. 455-460, 2001.

[3] Cordeiro. C, Abhyankar. S and Agrawal D.P, "Design and implementation of QoS-driven dynamic slot assignment and piconet partitioning algorithms over Bluetooth WPANs", *in Proceedings of IEEE INFOCOM*, pp. 1252–1263, 2004.

[4] Ching-Fang Hsu, Member, IEEE, and Shu-Ming Hsu, "An Adaptive Interpiconet Scheduling Algorithm Based on HOLD Mode in Bluetooth Scatternets", *IEEE Transactions on Vehicular Technology*, Vol. 57, No. 1, pp.475-489, 2008.

[5] Har-Shai. L, Kofman. R, Segall. A and Zussman. G," Load Adaptive Interpiconet Scheduling In Small-Scale Bluetooth Scatternets**",** *IEEE commication Magazine,* Vol. 42, No. 7, pp. 136-142, 2004.

[6] Jang-ping Sheu, Kuei-ping shih, Shin-Chih Tu"Traffic aware scheduling in bluetooth Scatternets", *IEEE Transactions on Mobile computing*, Vol. 5, No. 7, pp. 872-883, 2006.

[7] Johansson. P, Kapoor. R, Kazantzidis. M and Gerla. M, "Rendezvous scheduling in Bluetooth Scatternets" in *Proceedings of the IEEE Computer and Communications Societies*, pp. 318–324, 2002.

[8] Kim. J, Lim. Y, Kim. Y and J. S. Ma, "An adaptive Segmentation scheme for the Bluetooth-based wireless channel,"in *Proceedings of the 10th IEEE International Conference,* pp 440-445, 2001.

[9] Lin. T.Y and Tseng Y. C, "An adaptive sniff scheduling scheme for power saving in Bluetooth", *IEEE Wireless Communication*. Vol. 9, No. 6, pp. 92–103, 2002.

[10] Racz. A, Miklos. G, Kubinszky. F and Valko. A, "A pseudo random coordinated scheduling algorithm for Bluetooth scatternets", in *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pp.193–203, 2001.

[11] Raymond Y.L.Lee and Vincent W.S.Wong, "An Adaptive Scheduling Algorithm for Bluetooth Ad-Hoc Networks**",** *IEEE Transactions on Mobile Computing*, Vol. 5, pp. 3532-3537, 2005.

[12] Tan. G and Guttag. J, "A Locally Coordinated Scatternet Scheduling Algorithm", *in Proc. of IEEE Conference on Local Computer Networks*, pp. 340-346, 2002.