Analysis of FiWi Networks to Mitigate Packet Reordering

G.Sivakumar
M.E, Student, Communication Systems
K.L.N College of Engineering
Pottapalayam, Madurai

A.V.Ramprasad
Professor, Communication Systems
K.L.N College of Engineering
Pottapalayam, Madurai

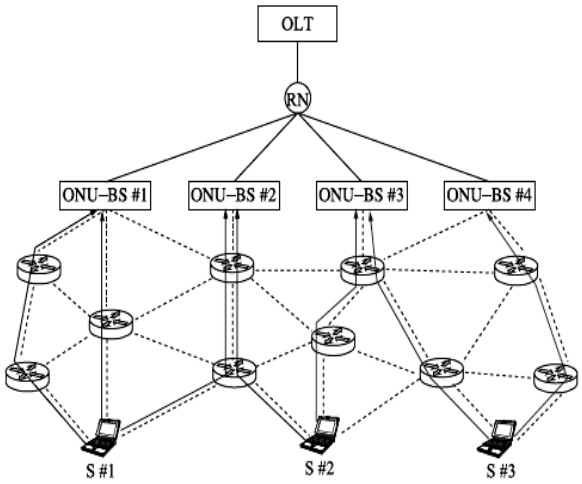
ABSTRACT
In an integrated fiber and wireless (FiWi) access networks, multipath routing may be applied in the wireless subnetwork to improve throughput. Because of different delays along multiple paths, packets may arrive at the destination out of order, which may cause TCP Performance degradation. As all traffic in a FiWi network is sent to the Optical line terminal (OLT), the OLT serves as a convergence node which naturally makes it possible to re-sequence packets at the OLT before they are sent to the internet. However the challenge is that OLT must re-sequence packets effectively with a very small delay to avoid a performance hit. To overcome this problem, Scheduling Algorithms (FIFO, Priority Queuing, DRR, and MDRR) at Optical Line Terminal (OLT) is used effectively to reduce Packet Reordering and to improve TCP Performance. Simulation results show that MDRR scheduling algorithm is effective in reducing the packet reordering.

General Terms
Multipath Routing, TCP Performance, Scheduling Algorithm.

Keywords
EPON, FiWi, WMN, Packet re-sequencing, MDRR, Packet Reordering.

1. INTRODUCTION
The Hybrid Fiber-Wireless (FiWi) Access Network integrates the passive optical networks (PON) and Wireless Mesh Networks (WMNs). FiWi networks aim at providing wired and wireless services over the same infrastructure simultaneously, thus potentially leading to major cost savings, high bandwidth and ubiquitous last mile internet access. A FiWi Network consists of EPON and Wireless Subnetwork as shown in Figure.1.

In a FiWi Network, in order to alleviate network congestion and improve throughput, packets of flow may be sent through multiple paths in wireless sub-network that are then sent to the optical line terminal (OLT) through different optical network units (ONUs) in the optical subnetwork Multipath routing is employed in order to increase the total network utilization, higher aggregate bandwidth, smaller end- to-end delay, and better load balancing [4].The OLT transmits the packets to their destination through core network. These packets, however, may be reordered at their destination due to delay variance along a duplicate ACK (dupack) for each out-of-order (OOD) segment. After receiving a number of Dupacks (typically three), the sender will assume that a segment has been lost and the network is congested. Consequently, the fast retransmit and fast recovery algorithm will be performed which causes additive increase in TCP's congestion widow size (cwnd).in a network where reordered packets are prevalent, TCP spuriously retransmits segments and keeps it



cwnd unnecessarily small, which would severely affect the TCP Performance.

A scheduling algorithm at the OLT, that aims to re-sequence the packets of each flow to facilitate in-order arrivals at the destination. Compared with re-sequencing at the end system, re-sequencing at the OLT has to be done fast enough so as not to introduce too much delay to the re-sequenced packets. The fast re-sequencing requirement implies that 100% in-order re-sequencing is impossible to achieve when re-sequening is conducted at the OLT.

2. RELATED WORK
The re-sequencing of packets is very important at the OLT because the OLT serves as a convergence node which naturally makes it possible to re-sequence packets at the OLT before they are send to the core network. However, some prior efforts are taken to re-sequence the packets at the intermediate nodes of the network. In packet re-sequencing [9] at OLT, for each flow the OLT maintains a priority queue system (TPQ) with two priority levels. The out of order packets to the OLT are put to the high priority level and otherwise to the low priority level, both levels are served in the first-in-first-out order. Although the time complexity of TPQ is *O* (1), the performance improvement highly depends on the output link capacity from the OLT to the backbone network reserved for each TCP flow.

To increase the throughput of TCP and to ensure possible in order packet delivery at the destination, new scheduling algorithm is proposed called Modified Deficit Round Robin Scheduling (MDRR) Algorithm. MDRR scheduling algorithm depends on the DRR fundamentals, in MDRR the quantum value given to the queues is based on the weight associated

with them. It provides better quality of service, and eventually leads to changing the throughput, latency, jitter and packet over flow at the queues.

3. TCP PERFORMANCE OVER MULTIPATH ROUTING

Transmission Control Protocol (TCP) provides reliable data transfer between node pairs. TCP receivers send an immediate acknowledgment (ACK) when a segment arrives. These ACKs are cumulative and acknowledge all in-order segment arrivals. TCP senders detect a packet loss by timer expiration or duplicate ACKs. In order to recover the lost packet fast, TCP senders retransmit the lost packet when three duplicate ACKs arrive (the fast retransmit algorithm). After the fast retransmission, the congestion avoidance is performed (the fast recovery algorithm).

TCP Performance over multiple paths is supposed to be enhanced, because the bottleneck bandwidth increases. Suppose that a pool of packets arrive dynamically from F different flows with packet reordering at the OLT where the OLT maintains a queue for each flow. Suppose that the time is partitioned into equal time slots where in each time slot at most one packet can be sent out from the OLT to the internet.

At the each time slot, the OLT needs to determine which flow (queue)'s packet should be sent and which packet from that selected flow (queue) should be sent. As duplicate ACKs (three dupacks) may trigger the fast retransmission and fast recovery, which will cause multiplicative decrease (and additive increase) in TCP's Congestion window size (*cwnd*).

It is important to avoid triggering three dupacks when we schedule packets. To achieve such a goal, at each time slot, a flow should be selected for transmission if it will most unlikely reduce the sender's cwnd and when a flow is selected for transmission, the packet with smallest sequence number in the queue should be scheduled. In other words, we can implement a min-heap queue for each flow and assure that the HOL packet of each flow has the smallest sequence number. On the other hand, fairness among flows shall also been considered when we schedule packets. Suppose that $P_{i,j}$ is the HOL packet of queue *i* at the *t.* Let $dack_i^{'}(t)$ denote the potential number if dupacks that may be caused by sending $P_{i,j}$. The impact of $dack_i^{'}(t)$ on the change of the sender's *cwnd* can be summarized follows.

- In-order delivery:

Case 1: $dack_i^{'}(t) = 0$, where $P_{i,j}$ is the expected packet. Transmitting $P_{i,j}$ will increase the sender's *cwnd* and allow the receiver to generate cumulative ACKs.

- Out-of- Order delivery :

Case 2: $dack_i^{'}(t) = 1$, To transmit $P_{i,j}$ will cause one dupack to be send to the sender, which, however, will not cause any change on the sender's *cwnd.*

Case 3: $dack_i^{'}(t) = 2$, Same to case 2.

Case 4: $dack_i^{'}(t) = 3$, To transmit $P_{i,j}$ will cause one dupack to be send to the sender, which will consequently trigger the fast retransmission and cause the reduction on the sender's *cwnd*.

Case 5: $dack_i^{'}(t) > 3$, TCP sender is now at the stage of fast recovery. To send $P_{i,j}$ will cause one dupack to be sent to the sender and increase the sender's *cwnd.*

As the sender's *cwnd* in both case 1 and case will be increased, such a flow *i* should have the highest priority to be scheduled for transmission at time slot t, Case 4 will cause the reduction on the sender's *cwnd*, thus, such a flow should be scheduled later with the hope that the expected packet will arrive at the queue. Case 2 and Case 3 will not cause the immediate change of the sender's *cwnd* and can be assigned with the priority between the highest priority and the lowest priority.

Let $p_i\ (t)$ be the priority of sending $P_{i,j}$, this is defined as follows:

$$p_i\ (t) = \begin{cases} 2 & \text{if } dack_i^{'}(t) = 1 \text{ or } dack_i^{'}(t) > 3 \\ 1 & \text{if } dack_i^{'}(t) = 1 \text{ or } dack_i^{'}(t) = 2 \\ 0 & \text{if } dack_i^{'}(t) = 3 \end{cases} \quad (1)$$

In order to mitigate the effect of packet reordering, the HOL packet of queue *i** with maximum $p_{i*}\ (t)$ will be scheduled, which enhances the chance of other queues with lower priority value to be resequenced. Apart from dupack, we also need to consider fairness among flows. Let $swait_i(t)$ be the time elapsed since last time when queue *i* is scheduled for transmission. If queue *i* not backlogged, we set $swait_i(t) = 0$. Thus, for the sake of fairness, the queue with highest $swait_i(t)$ should be scheduled.

Let $f_i\ (t)$ be the scheduling weight if flow *i* at time slot t. considering both priority from the perspective of potential change on the sender's *cwnd* and fairness, we define the following total order among flows: $f_i\ (t) > f_j\ (t)$ if $p_i\ (t) > p_j\ (t)$ or $p_i\ (t) = p_j\ (t)$ and $swait_i(t) > swait_j(t)$. In such a case, we may prefer to delay the transmission at current time slot and wait for the expected packet. Note that in the next time slot, with the new arrival packets to each flow, a new flow may be selected for transmission or the current flow will be selected for transmission again. On the other hand, if the expected packet is lost, a TCP timeout will eventually be triggered. In such a case, it is more desirable that we send the packet immediately to trigger fast retransmission.

4. PACKET SCHEDULING ALGORITHM AT OLT

The most challenging part of making the scheduling decision is to maintain the number of dupacks for each queue. The scheduling algorithm at optical line terminal, which aims to re-sequence packets as much as possible.

Assume that queue i has sent out $P_{i,1}$ and $dack_i^{'}(t) = 0$. The OLT is now expecting $P_{i,2}$. When packet $P_{i,3}$ becomes the HOL packet, suppose queue i is scheduled for transmission and packet $P_{i,3}$ is sent out, $dack_i^{'}(t)$ becomes 1. In the next time slot, suppose the packet $P_{i,5}$ becomes the head of queue i. in this case, the current packet's seqno is even higher than the highest seqno sent so far for this queue. Suppose that this packet is sent out immediately, $dack_i^{'}(t)$ becomes 2. In the next time slot, if the expected packet $P_{i,2}$ arrives at the OLT and is sent out immediately. The expected seqno will be 4 as packet $P_{i,3}$ has been sent out from the OLT. This situation

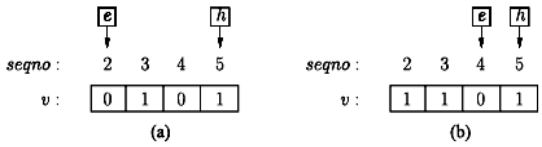


Figure 2. A bit_vector for reordering the OOD packet delivering

Includes that in order to update the expected seqno, we have to record which packets have been sent among the packets with seqno between current expected seqno and the highest seqno.

We use a bit_vector v to record which packets between the expected one and the highest seqno have been sent out (Fig 2) where v[i] = 0 indicates that the corresponding packet has not been sent out from the OLT, and vice versa. As shown in (Fig 2 (a)), the expected packet seqno is 2, the highest packet's seqno sent so far is 5, packet 3 has been sent out. Thus, when the expected packet $P_{i,2}$ arrives, we can immediately obtain that the next expected seqno is 4 (Fig 2 (b)).this shows that, in order to Maintain the number of dupacks,we need the information of the expected seqno,the highest seqno sent so far, and a vector for recording information about out-of-order packet delivering. We now formally present how to maintain the expected seqno, highest seqno, the number of committed dupaks, and the number of potential dupacks for each queue.

The OLT maintains a quadruplet {*e*, *dack*, $dack'$,*h*} and a variable (*swait*) for each flow *i* , where e denotes the expected seqno, h denotes the highest *seqno* of packet which has been sent, *dack* denotes the number of committed dupacks, $dack'$ denotes the number of potential dupacks and *swait* denotes the time elapsed since flow *i* is scheduled for transmission last time. Each queue is maintained as a min-heap queue where the HOL packet has the minimum seqno.

Initially, {*e*=0, *dack*=0, $dack'$=0, *h*= -1) and *swait* =0 for each queue, which indicates that no packet has been sent for this flow and the packet with *seqno*=0 is expected to sent next.

Whenever a packet enters a queue, min-heap insertion Operation will be conducted at the queue. If the HOL packet of the queue remains to be the same, no update is necessary. Suppose that the current packet becomes the Head of line packet and its *seqno* is *j*. We need to see how $dack'$ will be changed if the HOL packet of this queue is scheduled next.

4.1 Scheduling Algorithms

Packet scheduling is necessary when multiple packets compete for a common outgoing link. The packet scheduling is nothing but a shared transmission resources should be intentionally assigned to some users packets to appropriate shared resources to achieve some performance guarantee is so called packet scheduling. Scheduling is usually done for load balance and to achieve quality of services (QOS). In FiWi networks the scheduling algorithm, take place at optical line terminal.

In this paper the scheduling algorithms are used to analysis the FiWi Networks. Fundamental scheduling algorithms are FIFO, TPQ DRR, and our proposed scheduling algorithm is MDRR.

First traditionally, the First Come First Served (FCFS) scheme follows the First in First out (FIFO) memory stack, as each process becomes ready, it joins the ready queue. When the current running process ceases to execute, the oldest process in the ready queue is selected for running. That is first entered process among the available process in the ready queue. The average waiting time for FCFS is often quite long.

The re-sequencing algorithm (TPQ) is based on a priority queuing system. For each flow, a priority Queuing system with two first in, first out (FIFO) queues is maintained at the OLT. Upon the arrival of a packet, if a packet with a higher sequence number has arrived earlier, the current packet will be placed in the low priority queue; otherwise it will be placed in the low priority queue.

Deficit round robin (DRR), also Deficit weighted round robin (DWRR) is a modified weighted round robin scheduling discipline. It can handle packets of variable size without knowing their mean size. A maximum packet size number is subtracted from the packet length, and packets that exceed that number are held back until the next visit of the scheduler. The DRR scheduling algorithm maintains a quantum value that defines the total number of credits for each Class of Service (CoS) queue and a credit counter that is decremented each time a byte is taken from the queue for transmission. The purpose of the credit counter is to track the use of bandwidth by a Class of Service (CoS) queue relative to the amount of bandwidth that has been allocated to the queue. Compared with Fair queuing (FQ) scheduler that has complexity of O (log(n)) (n is the number of active flows), the complexity of DRR is O(1).

5. MODIFIED DEFICIT ROUND ROBIN

MDRR scheduling is an extension of the Deficit round robin (DRR) scheduling scheme. The algorithm depends on the DRR scheduling fundaments to a great extent, however, in MDRR the quantum value given to the queues is based on the weight associated with them, as indicated in Equation 2.

$$q = mtu + 512*w; \quad (2)$$

Where, q= quantum, w= weight, and mtu=Maximum transmission unit; Maximum Transmission Unit (MTU) is the maximum packet size that a queue may deliver. Note that, since the MTU is a constant number for a given system, quantum value and weight are therefore directly proportional and hereafter could be used interchangeably. The reason of including the MTU parameter in equation 2 is to ensure that the quantum to be delivered to the intended queue at least enables the queue to transmit one packet. Since if no packet was transmitted in a round this results in the increase of the operational complexity.

The MDRR scheduling scheme adds a Priority Queue (PQ) into consideration with DRR. A Priority Queuing scheme isolates high demanding flows from the rest of the other flows for the reason of better quality of service provisioning. This is illustrated as shown in Figure 3.According to the mode of serving the Priority Queue, there are mainly two types of MDRR schemes:

- Alternate mode: In this mode the high priority queue is served in between every other queue. For instance the scheduling sequence may be follows: {PQ, Q1, PQ, Q2, PQ Q3, PQ, and Q4}.
- Strict Priority mode: here the high priority queue is served whenever there is backlog. After completely transmitting, all its packets then the other queues are served. However, as soon as packets backlogged again in the high priority queue, the scheduler transmits the packet currently being served and moves back to the high priority queue.

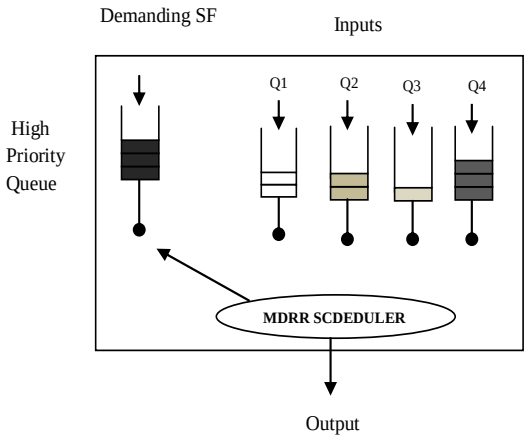


Figure 3. MDRR Scheduler

6. PERFORMANCE EVALUATION

We present our simulation results and compare our MDRR algorithm with other scheduling algorithms. We evaluate the performance of our proposed MDRR scheduling algorithm at the OLT.

6.1 Experimental Setup

We evaluate the performance of our algorithm using NS-2 (version 2.27). Figure 4 shows the simulation topology. Each sending packets to the OLT through 2 paths in a round robin fashion. Each path from the source to the OLT can provide 20 Mbps link capacity.

The OLT is connected to the internet through an Ethernet. As mentioned earlier, packets of a flow may arrive at the OLT through different paths. Due to different path's delay, OOD packet arrivals will be produced. In order to simulate the delay difference between paths, for the two paths from each source to the ONU, one's delay is 2 ms and other is 2 + d ms where d varies from 0 to 50 ms. A large d will introduce more variation in the path delay, thus increasing the degree of packet reordering. We develop the EPON with transmission speed 1 GB/s. The round-trip propagation delay for each ONU is assumed to be random (uniform). Every ONU has a buffer size is set to be 10Mbytes.

We use TCP/Reno as the agent for TCP connection. TCP/Reno is chosen for its implementation of fast retransmission and fast recovery. The TCP throughput B_{TCP} can be expressed as

$$B_{TCP} = \left(B_i\ ,\ \frac{W_{max}}{RTT}\ .MSS\right), \qquad (3)$$

Where B_i is the bandwidth for source *i*,W_{max} is the maximum cwnd size, and *MSS* is the maximum segment size.

6.2 Simulation Results

We compare the performance of our proposed MDRR Scheduling algorithm with other scheduling algorithms such as FIFO, DRR, and TPQ (Re-sequencing Algorithm). We use goodput, which is defined as the number of packets successfully received and acknowledged by the receiver, excluding retransmissions, as a performance metric to compare our proposed packet scheduling algorithm with other packet scheduling algorithms.

Firstly, the buffer size at the OLT is set to be large enough to ensure that no packet drops at the OLT Figure 5 shows the average goodput of all flows when d varies between 0 ms and 10ms at intervals of 2 ms.we can see that larger d lead to lower goodput. The goodput of FIFO, DRR decreases sharply when d increase.

Fig. 6 shows that the packets experience much longer queuing delay at the OLT in FIFO than other schedulers; the FIFO is not practical to be used as a re-sequencing algorithm at the intermediate nodes. Our proposed MDRR scheduling algorithm will experience much less queuing delay at the OLT when the delay difference among multiple paths is moderate. Because the maximum queuing delay is proportional to the buffer size, to show how much buffer size is required at the OLT each scheduler, we set d to 20ms and simulate the goodput when buffer size varies from 16 to 1024 Kbytes with exponential scale. Here, we adopt the shared buffer scheme, i.e., all flows share a common buffer pool in the OLT. When overflow occurs, we use a pointer to drop the tail of each queue periodically.

With increase of buffer size fewer packets are dropped. Hence, the goodput of upstream TCP traffic also increases. From Fig. 7 we can see that the goodput of MDRR indeed increases when compared with FIFO, TPQ and DRR.

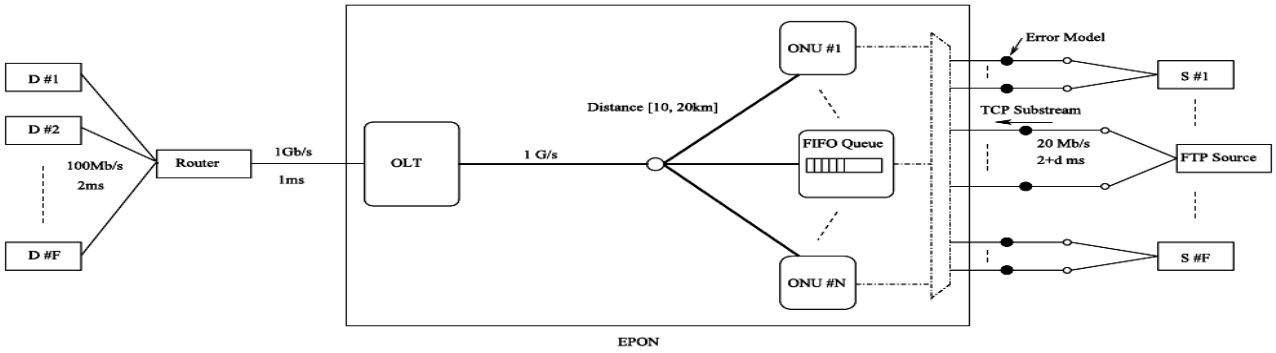


Figure 4. Simulation Network Topology

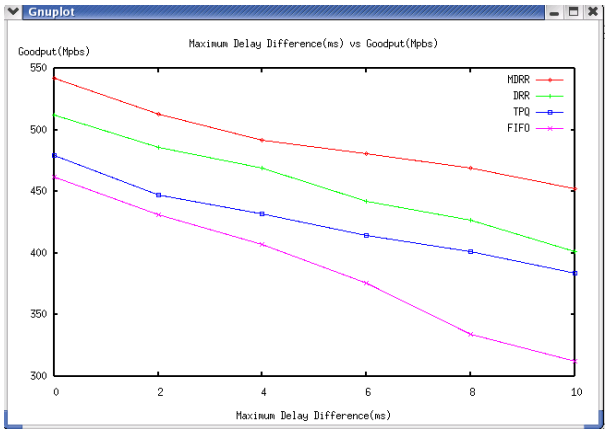


Figure 5. Goodput Versus Maximum delay difference.

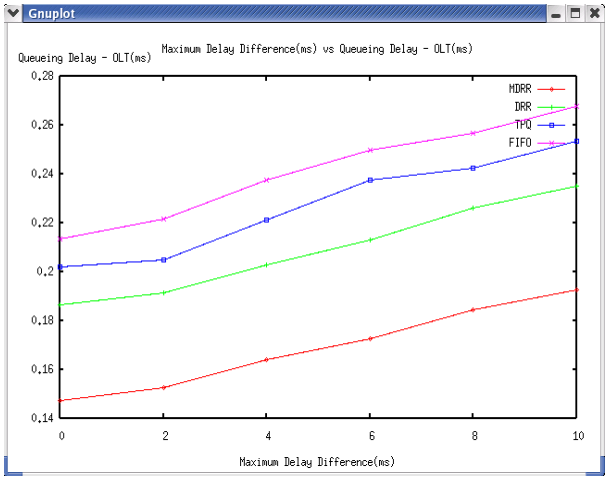


Figure 6. Queuing delay at the OLT Versus maximum delay difference

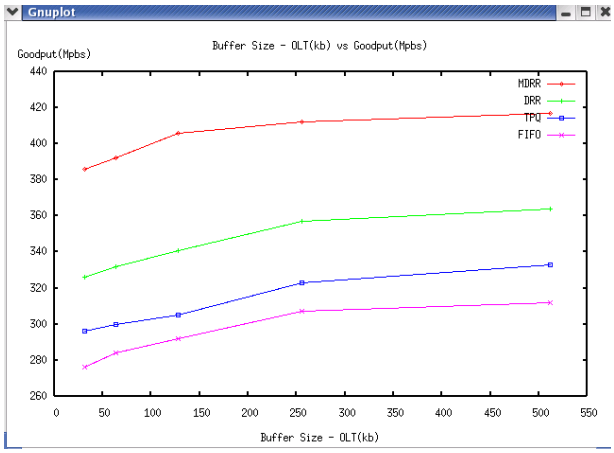


Figure 7. Goodput versus buffer size in the OLT

Form Figure 7, we can see that the goodput of MDRR scheduling algorithm is increase with the buffer size while compared with other FIFO, TPQ and DRR scheduling algorithms.

Figure 8. Shows the achieved Packet delivery ratio of those scheduling algorithm when simulation time varies from 50 to 250 seconds.

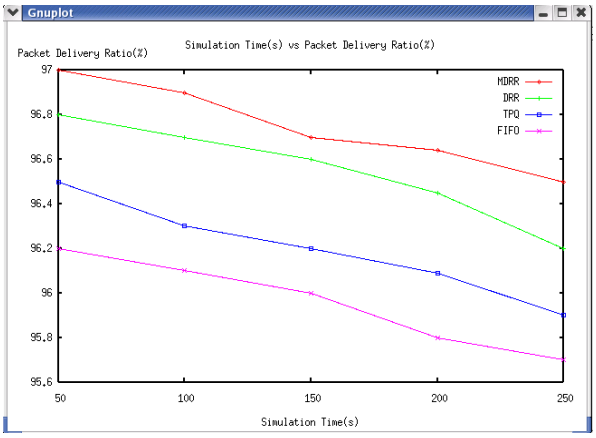


Figure 8. Packet delivery Ratio versus simulation time

7. CONCLUSION

The integration of a Passive Optical Network and a Wireless Mesh Network makes the OLT as a convergence node; in a FiWi network multipath routing is adopted in order to mitigate network congestion. In this paper, we propose a MDRR scheduling algorithm at the OLT to resequence packets while providing fairness. Simulation results shows that the proposed packet scheduling algorithm (MDRR) is efficient in reducing the effect of packet reordering, assuring fairness among the different flows and reducing the required buffer size in the OLT. Compared with other scheduling algorithms, our proposed scheduling algorithm provides the scalable solution to resequence the packets.

8. ACKNOWLEDGEMENT

G.Sivakumar thanks to Prof. V. Kejalakshmi (Head of the Electronics and Communication Department, K.L.N. College of Engineering), Prof. R. Jayanthi (ECE Department, K.L.N. College of Engineering), I am grateful to Dr.A.V.Ramprasad (Principal, K.L.N. College of Engineering) for allowing me to use some of their published research results.. Finally, I am indebted to K.L.N. College of Engineering for encourage my research work.

9. REFERENCES

[1] Shiliang Li, Jianoing Wang, Chumming Qiao, and Yinlong Xu, "Mitigating Packet Reordering in FiWi Networks", J.opt.commun.Netw vol.3, no.2, pp.134-144, Feb 2011.

[2] S. Sarkar, S. Dixit, and B. Mukherjee, "Hybrid wireless-optical broadband-access network (WOBAN): a review of relevant chal- lenges," J. Lightwave Technol., vol. 25, no. 11, pp. 3329–3340, Nov. 2007.

[3] M. K. Marina and S. R. Das, "Ad hoc on-demand multipath distance vector routing," ACM SIGMOBILE Mob. Comput. Com- mun. Rev., vol. 6, no. 3, pp. 92–93, 2002.

[4] K. Leung and V. Li, "Flow assignment and packet scheduling for multipath routing," J. Commun. Netw., vol. 5, no. 3, pp. 230–239,2003.

[5] M. Zhang, B. Karp, S. Floyd, and L. Peterson, "RR-TCP: a reordering-robust TCP with DSACK," in Proc. 11th IEEE Int. Conf. on Network Protocols, pp. 95–106, 2003.

[6] M.Shreedar and G.Varghese, "Efficient fair queuing

using deficit round robin", IEEE/ACM Trans.Netw,vol. 4, no. 3,pp. 375-385,1996.

[7] Y. Lee, I. Park, and Y. Choi, "Improving TCP Performance in multipath packet forwarding networks," J. Commun. Netw., vol. 4, pp. 148–157, 2002.

[8] B. Radunovi, C. Gkantsidis, D. Gunawardena, and P. Key, "Horizon: balancing TCP over multiple paths in wireless mesh network," in Proc. 14th ACM Int. Conf. on Mobile Computing and Networking, San Francisco, CA, pp. 247–258,2008.

[9] J. Wang, K. Wu, S. Li, and C. Qiao, "Performance modeling and analysis of multipath routing in integrated fiber-wireless networks," in Proc. IEEE INFOCOM, pp. 1–5,2010

[10] J. Lane and A. Nakao , "Best – effort layer packet reorder- ing in support of multipath overlay packet dispersion," in IEEE Global Telecommunications Conf., pp. 1–6,2008.

[11] M. P. Mcgarry, M. Maier, and M. Reisslein, "Ethernet PONS: a survey of dynamic bandwidth allocation (DBA) algorithms," IEEE Commun. Mag., vol. 42, pp. 8–15, 2004.

[12] H. Ikeda and K. Kitayama, "Dynamic bandwidth allocation with adaptive polling cycle for maximized TCP through- put in 10G-EPON," J. Lightwave Technol., vol. 27, no. 23, pp. 5508–5516, Dec. 2009.